Université de Montréal

# S-JET : Une nouvelle conception pour la gestion de réservation pour l'architecture des réseaux OBS

par
Eloim Gutierrez Cabrera

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de Maître ès sciences (M.Sc.)
en Informatique

Avril, 2005



Université de Montréal
Faculté des études supérieures

Ce mémoire intitulé:

**S-JET : Une nouvelle conception pour la gestion de réservation pour l'architecture des réseaux OBS**

présenté par:

Eloim Gutierrez Cabrera

a été évalué par un jury composé des personnes suivantes:

| | |
|---|---|
| Jean-Yves Potvin, | président-rapporteur |
| Felisa Vázquez-Abad, | directeur de recherche |
| Brunilde Sansó, | codirecteur |
| André Girard, | membre du jury |

**Mémoire accepté le:** 27/07/05

# Sommaire

Nous proposons S-JET, un algorithme efficace pour la gestion de réservation pour l'architecture des réseaux OBS. La méthode de réservation est basée sur la discrétisation de la rafale ainsi que du temps de réservation. En utilisant des opérations booléennes l'algorithme détermine s'il y un intervalle libre assez grand pour allouer une demande d'une réservation dans un des canaux au port de sortie. Lorsque plusieurs possibilités existent, l'algorithme trouve, avec une simple comparaison par canal, le canal qui produira le moindre gaspillage.

Les algorithmes Horizon et JET sont deux algorithmes de gestion de réservation largement connus pour les réseaux d'OBS. Plusieurs études ont été effectuées pour évaluer leur probabilité de blocage. En utilisant cette approche, nous avons constaté que la probabilité de blocage de l'algorithme S-JET tend vers la performance de l'algorithme JET. Cette performance est obtenue sans considérer la complexité informatique du modèle des réseaux OBS qui affecte les décalages qui arrivent au noeud. Lorsqu'on incorpore cette complexité, la performance de JET se détériore. S-JET a été conçu pour incorporer le meilleur des deux algorithmes : la vitesse de l'algorithme Horizon et l'utilisation efficace des longueurs d'onde de l'algorithme JET.

D'autre part, à partir de nos expériences préliminaires, il est évident que S-JET

est capable de faire une différentiation selon la taille de la rafale, ce qui entraînerait une priorité implicite aux rafales plus petites tout en conservant dans le pire des cas, une probabilité de blocage toujours plus basse que celle de l'algorithme Horizon. Nous conjecturons qu'un système de priorités basé sur la taille peut être une alternative intéressante aux méthodes de différentiation de services proposées pour OBS.

**Mot-clés :** Réseaux à commutation de rafales optiques, design de réseau optique, algorithmes d'ordonnancement de rafales.

# Abstract

We propose S-JET, an efficient reservation scheduling algorithm for OBS switches. The method is based on the discretization of the burst and the reservation time frame combined with simple Boolean operations to determine if there is space to allocate a request for a reservation in one of the outgoing channels. When several possibilities exist, the algorithm finds, with a single comparison per channel, the channel allocation that will produce the smallest gap.

Horizon and JET are two widely known reservation algorithms proposed for OBS networks. Several studies have been made to evaluate their blocking probability. When using this approach, we found that the behaviour of S-JET tends toward the performance of JET. This performance is achieved without considering the OBS model feature, where the computational complexity may affect the incoming offsets. When this is incorporated in the simulation, the performance of JET deteriorates. S-JET has been designed to incorporate the best of both algorithms : the low processing time of the Horizon algorithm and the efficient use of wavelengths of JET.

In addition, from our preliminary experiments reported here, it is apparent that S-JET is capable of differentiation by size. This in turn yields an implicit priority to smaller bursts, while keeping the worst case blocking still lower than the corresponding Horizon algorithm. We conjecture that a system based on priorities per size

may actually be an ideal alternative to currently proposed differentiation methods in OBS. This is due to its simplicity and because the main control actions are kept at the edge router, making the controlled dynamics very fast and efficient in the interior of the core OBS network.

**Keywords :** OBS networks, optical network design, fiber optics and optical communications, burst scheduling algorithms.

# Table des matières

**Sommaire** iii

**Abstract** v

**Table des matières** vii

**Liste des figures** x

**Glossaire des traductions** xii

**Liste des tableaux** xiv

**Dédicace** xv

**Remerciements** 1

**Introduction** 2

**1 Les réseaux OBS** 6

1.1 Média de transmission . . . . . 6

1.2 Commutation et traitement de données . . . . . 8

1.2.1 Commutation optique . . . . . 10

1.3 Architecture du réseau . . . . . 11
1.3.1 En-tête de contrôle . . . . . 12
1.3.2 Liens et canaux . . . . . 13
1.3.3 Assemblage de rafales . . . . . 13
1.3.4 Commutateur OBS . . . . . 14
1.3.5 Mécanisme d'envoi de rafales . . . . . 15
1.4 Contexte . . . . . 16

**2 Modèles de réservation dans les commutateurs 17**

2.1 Algorithmes de réservation . . . . . 17
2.1.1 Procédure de réservation avec Horizon . . . . . 20
2.1.2 Procédure de réservation avec JET . . . . . 22
2.2 Procédure de réservation dans S-JET . . . . . 25
2.3 Autres approches . . . . . 29

**3 Implantation de S-JET 31**

3.1 Structure des données . . . . . 32
3.2 Procédure de réservation . . . . . 33
3.3 Une procédure rapide pour choisir le meilleur canal . . . . . 35

**4 Analyse des simulations 44**

4.1 Modèle de simulation . . . . . 45
4.1.1 1 : Rafales à taille constante . . . . . 45
4.1.2 2 : Rafales de taille variable . . . . . 46
4.2 Calcul de $\Gamma$ et $M$ . . . . . 46
4.3 Complexité des algorithmes . . . . . 48
4.4 Comparaison de la probabilité de perte . . . . . 50

4.4.1 Probabilité de perte avec rafales de taille constante . . . . . 51
4.4.2 Probabilité de perte avec rafales de taille variable . . . . . 52
4.5 Observations . . . . . 58

**Conclusion** **61**

**Bibliographie** **63**

**A T-SIM : simulateur de réseaux optiques** **67**
A.1 Introduction . . . . . 67
A.2 Fonctionnement de T-SIM . . . . . 68
A.2.1 Le noyau de T-SIM . . . . . 68
A.2.2 Module OBS . . . . . 69
A.2.3 Exécution d'une simulation . . . . . 74
A.3 Conclusion . . . . . 77

# Liste des figures


1.1 Vue d'une fibre . . . . . 7

1.2 Multiplexation de plusieurs signaux dans une fibre optique . . . . . 8

1.3 Réseau OBS . . . . . 12

1.4 Décalage . . . . . 13

1.5 Assemblage de rafales . . . . . 14

1.6 Commutateur OBS . . . . . 15

2.1 Mise à jour du décalage . . . . . 18

2.2 Modèle de réservation avec Horizon . . . . . 21

2.3 Modèle de réservation avec JET . . . . . 23

3.1 Structure des données, avec $\sigma = 64$ . . . . . 33

3.2 Schème de réservation avec S-JET . . . . . 34

3.3 Une rafale qui occupe deux cellules consécutives . . . . . 35

3.4 Sélection du canal avec le plus petit nombre de cases vides . . . . . 37

3.5 Blocage d'une réservation avec S-JET *myope* . . . . . 39

3.6 Blocage d'une réservation avec S-JET *théorique* . . . . . 40

3.7 Une source additionnelle de blocage pour S-JET . . . . . 41

4.1 Performance de S-JET en utilisant différentes valeurs pour $M$ . . . . 48

4.2 Probabilité de blocage pour Horizon, JET et S-JET . . . . . 52
4.3 Probabilité de perte globale de Horizon et S-JET, scénario 1 . . . . . 54
4.4 Probabilité de perte par taille S-JET . . . . . 55
4.5 Probabilité de perte globale Horizon et S-JET, scénario 2 . . . . . 56
4.6 Probabilité de perte par taille S-JET, scénario 2 . . . . . 56
4.7 Probabilité de perte globale de Horizon et S-JET, scénario 3 . . . . . 57
4.8 Probabilité de perte par taille S-JET, scénario 3 . . . . . 58

A.1 Structure de T-SIM . . . . . 68
A.2 Réseau exemple . . . . . 70
A.3 Classes T-SIM (1) . . . . . 78
A.4 Classes T-SIM (2) . . . . . 79
A.5 Classes T-SIM (3) . . . . . 80
A.6 Classes OBS (1) . . . . . 81
A.7 Classes OBS (2) . . . . . 82
A.8 Classes OBS (3) . . . . . 83

# Liste des notations et de symboles

**Notation**

| | |
|---|---|
| $b_n$ | Taille de la $n^e$ rafale. |
| $\bar{b}$ | Taille maximal de la rafale. |
| $C_n$ | Ensemble de canaux candidats. |
| $c_n$ | Canal où la rafale sera allouée. |
| $e_l$ | Fin de l'intervalle de transmission. |
| $H$ | Nombre maximal de sauts dans une route. |
| $M$ | Nombre de cases nécessaires pour représenter une rafale. |
| $\mathcal{T}$ | Horizon de la liste de réservation. |
| $\Gamma$ | Taille de la case. |
| $\delta$ | Décalage. |
| $\tau$ | Temps de traitement pris par le contrôleur. |
| $s_l$ | Début de l'intervalle de transmission. |

**Glossaire des traductions**

| | |
|---|---|
| Burst | Rafale |
| Control header | En-tête de contrôle |
| Dense Wavelength Division Multiplexing | Multiplexage en longueur d'onde dense |
| Optical Circuit Switching | Commutation optique de circuits |
| Optical Cross Connect | Matrice optique de commutation |
| Optical Packet Switching | Commutation optique de paquets |
| Optical Burst Switched Network | Réseau à commutation de rafales optiques |
| Offset | Décalage |
| Slot | Case |
| Wavelength Division Multiplexing | Multiplexage en longueur d'onde |

# Liste des tableaux

4.1 Paramètres de la simulation . . . . . 47

4.2 Temps de traitement en tics de CPU requis par les algorithmes . . . . 49

4.3 Temps de traitement $\tau$ (en microsecondes) requis par les algorithmes 50

4.4 Taille de la rafale . . . . . 53

A la memoria de mi padre

# Remerciements

Je tiens à remercier toutes les personnes qui ont contribué de près ou de loin à ce travail. En particulier, je tiens à remercier mes directrices de recherche madame Felisa Vázquez-Abad (University of Melbourne) et madame Brunilde Sansò (École Polytechnique). Leurs directives et leur conseils ainsi que leur critiques, m'ont beaucoup inspiré et m'ont permis d'atteindre mes objectifs.

Un merci à Jolyon Whyte, il a su m'expliquer les subtilités que je n'avais pas bien saisi. J'ai apprécié beaucoup son aide lors de son séjour ici à Montréal et mon séjour à Melbourne.

Je désire faire mes remerciements à ma mère, mes frères ainsi que ma soeur qui m'ont supporté malgré la distance.

Finalement, je remercie mes amis qui ont été près de moi pendant la durée de mes études, en particulière Irina, Kim, Katerine, Nabil, Hichem et Joëlle.

# Introduction

Plusieurs mécanismes de réservation des canaux pour les réseaux de commutation de rafales ont été présentés dans la littérature. Ces mécanismes sont utilisés pour trouver un intervalle libre suffisamment grand pour allouer la rafale au port de sortie des commutateurs optiques. Dans le cas où plusieurs canaux sont disponibles pour allouer la réservation, le canal offrant la meilleure utilisation de ressources, c'est-à-dire, moins de gaspillage de bande passante, est choisi. La probabilité de perte ainsi que l'utilisation de ressources sont des critères employés pour évaluer la performance d'un mécanisme. En même temps, d'autres facteurs comme le délai de bout en bout ou la gigue jouent un rôle important dans la sélection du mécanisme de réservation. Il arrive que pour certains de ces mécanismes, la probabilité de perte de rafales est faible, sans tenir compte leur complexité informatique. Une fois que celle-ci est considérée, ils deviennent moins performants. Aussi, il existe des algorithmes ayant une probabilité de perte plus élevée mais avec une complexité informatique moindre.

Ce mémoire propose des idées originales qui rendent la gestion de réservation plus efficace pour les réseaux de commutation de rafales optiques, de sorte que la probabilité de perte de rafales dans les commutateurs est réduite sans encourir une complexité informatique élevée.

Ces dernières années, la conception d'un nouveau paradigme de commutation optique a fait l'objet d'études de la part de chercheurs. Dans la *commutation optique de paquets* (OPS), les paquets IP sont lus à chaque commutateur afin de déterminer la route à suivre par le paquet. Ceci entraîne la mise en mémoire tampon sous forme optique de la charge utile de chaque paquet lorsque l'en-tête correspondant est lu et traité par le commutateur optique sous forme électronique. Les inconvénients de cette approche sont reliés au matériel. Jusqu'à maintenant, les commutateurs optiques ne sont pas capables de traiter les données à la même vitesse que celle à laquelle les paquets sont transportés par la fibre optique [4]. Une alternative à l'approche OPS est la *commutation optique de circuits* (OCS). Ici, un circuit optique est établi à l'avance entre le commutateur d'origine et le commutateur destination. Ceci permet la transmission entièrement optique des paquets à travers le circuit. Dans cette approche, les inconvénients se présentent du côté du gaspillage de la bande passante ainsi que par le délai additionnel causé par l'établissement du circuit.

La *commutation de rafales optique* (OBS) combine les avantages de la commutation optique de circuits et la commutation optique de paquets, tout en évitant les inconvénients mentionnés ci-dessus. OBS favorise l'emploi de la commutation optique pour l'acheminement de données, mais elle considère également la flexibilité de l'électronique en matière de contrôle. De plus, OBS accomplit une utilisation efficace du spectre optique disponible dans les réseaux utilisant le multiplexage en longueur d'onde.

Le système de réservation mis en place à chaque commutateur rend disponible un des canaux sur le port de sortie au moment de l'arrivée de la rafale. JET [14] et Horizon [22] sont deux systèmes de réservation utilisés en OBS à cette fin. L'al-

gorithme Horizon est un mécanisme de réservation simple à implanter. Sa logique implique la visualisation de l'horizon de chaque canal dans le port de sortie, afin de déterminer la disponibilité d'allouer la réservation. L'algorithme JET pour sa part, exploite les intervalles libres entre les réservations déjà sur place. Ceci le rend plus performant en terme de probabilité de perte de rafales, mais si nous considérons sa complexité informatique, alors il devient un mauvais choix comme algorithme de réservation. Nous présentons S-JET, un algorithme conçu pour la gestion de réservation de canaux pour les réseaux OBS. Notre implantation profite des avantages des algorithmes Horizon et JET afin d'offrir un algorithme ayant une performance semblable a celle de JET mais avec une simplicité informatique similaire à celle de Horizon.

La contribution de ce mémoire est l'application de la discrétisation des canaux donnant lieu à la réservation de rafales en utilisant des opérations booléennes pour accélérer le temps de traitement au contrôleur. Les résultats performants de l'algorithme S-JET nous ont permis d'obtenir un brevet provisoire [19].

Le texte est divisé comme suit : dans la première partie de ce mémoire, nous présentons l'état de l'art des réseaux OBS. Nous y abordons le problème du trafic croissant sur les réseaux en général et comment les réseaux optiques sont venus résoudre ce problème. Par la suite, nous présentons l'architecture de réseau de commutation de rafales optiques. La deuxième partie (Chapitre 2) présente les modèles de réservation dans les commutateurs optiques. Nous y exposons en particulier les méthodes JET et Horizon ainsi que notre nouvelle méthode : S-JET. Nous expliquons la base de notre algorithme. La troisième partie (Chapitre 3) développe la méthode proposée. C'est dans ce chapitre que nous expliquons la mise en oeuvre de

S-JET. La quatrième partie (Chapitre 4) porte sur l'analyse des résultats numériques obtenus. Nous décrivons au préalable les modèles utilisés pour nos tests ainsi que le calcul de certains paramètres. Par la suite, nous présentons les tests réalisés en utilisant des rafales de tailles constantes et de tailles variables. Puis, nous effectuons la comparaison des résultats obtenus. Finalement, nous faisons part de la conclusion sur le travail accompli dans ce mémoire et nous proposons des idées pour des travaux futurs.

# Chapitre 1

# Les réseaux OBS

La grande acceptation de l'Internet dans toutes les sphères de la vie humaine a fait de lui un système largement utilisé dans notre société. Des estimations indiquent que l'utilisation de la bande passante double chaque six à douze mois [10]. À ce taux de croissance, le design d'une nouvelle infrastructure de réseau capable de faire face à un tel trafic est devenu un enjeu pour l'industrie de la télécommunication et les constructeurs de matériel informatique. Ce chapitre présente l'état de l'art des réseaux de commutation de rafales. Dans les sections 1.1 et 1.2 nous présentons les médias de transmission utilisés à cet effet ainsi que deux techniques de commutation de données par fibre optique. Ensuite, dans la section 1.3, nous présentons un aperçu de l'architecture des réseaux de commutation de rafales. Nous introduisons aussi les concepts de base utilisés dans ce mémoire.

## 1.1 Média de transmission

Les réseaux à câble électrique de cuivre, coaxial et paire torsadée, sont capables d'atteindre des vitesses de transmission qui atteignent les gigabits par seconde. Tou-

tefois, la demande élevée de bande passante a motivé la recherche d'autres médias de transmission plus performants. L'arrivée de la fibre optique a été un grand pas pour acheminer des grandes quantités de données par les réseaux [1, 27].

En tant que media de transmission, la fibre optique a plusieurs avantages par rapport au cuivre. L'atténuation est inférieure, les taux de transfert plus élevé et il n'y a aucune interférence électromagnétique. En plus, la fibre est plus légère et plus résistante que le cuivre. Naturellement, la fibre a également quelques inconvénients, tels que la dispersion, la réfraction non linéaire ou l'atténuation. Cependant, comparé à d'autres médias de transmission, la fibre est une alternative attrayante. En fait, dernièrement l'utilisation de la fibre optique a été préférée aux fils électriques grâce à ses nombreux avantages [20]. La fibre est un long et mince fil de verre très pur d'un diamètre de 8 à 10 microns, similaire à un cheveu humain, par lequel un faisceau lumineux transporte des données. La fibre est faite de deux parties, le coeur et la gaine. C'est cette dernière qui confine l'énergie lumineuse à l'intérieur du coeur. Un revêtement de protection couvre ces deux parties internes (fig. 1.1).

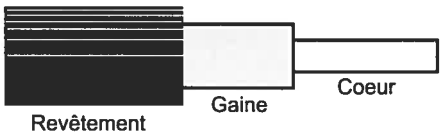



FIG. 1.1 – Vue d'une fibre

Le *multiplexage en longueur d'onde*, WDM, est la version optique du multiplexage par répartition en fréquence (FDM). Cette technologie, qui présente une très haute fiabilité, consiste à transporter plusieurs signaux optiques sur une seule fibre. L'idée de base est de diviser le spectre optique en sous-canaux où chaque sous-canal a une longueur d'onde différente (fig. 1.2). De cette manière, WDM fournit plusieurs

canaux virtuels sur une seule fibre physique. Les systèmes WDM actuels utilisent jusqu'à 32 longueurs d'onde par fibre à une vitesse de 2.5 gigabit/s [16].

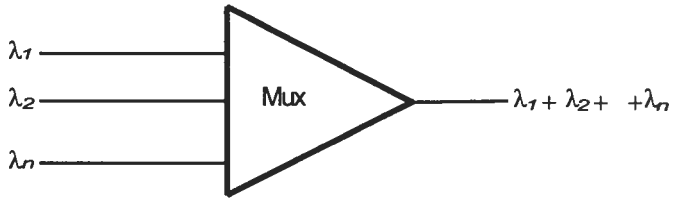



FIG. 1.2 – Multiplexation de plusieurs signaux dans une fibre optique

Aujourd'hui, WDM est l'alternative la plus populaire pour multiplexer plusieurs signaux dans le domaine optique. Ses avantages principaux sont sa transparence, son extensibilité et sa flexibilité : les lignes existantes de fibre peuvent être mises à jour en utilisant la technique WDM.

Les améliorations faites à l'architecture WDM ont permis son développement à un point tel qu'il est aujourd'hui possible de multiplexer jusqu'à 128 canaux dans une seule fibre. Cette nouvelle version est connue comme *multiplexage en longueur d'onde dense* (DWDM) et a rendu possible une utilisation plus efficace de la capacité de bande passante de la fibre optique sans présenter des coûts élevés.

## 1.2 Commutation et traitement de données

Un commutateur est un dispositif utilisé pour commuter des signaux des ports d'entrée aux ports de sortie. Une façon simple de les classifier est par commutateurs optiques et par commutateurs électroniques. Les commutateurs électroniques sont des dispositifs plus mûrs comparés aux commutateurs optiques et leur implantation

est plus facile. Étant donné que la technologie courante ne permet pas la manipulation directe des données sous forme optique, la conversion opto-électronique prendre place lorsque des données doivent être traités dans un commutateur électronique. Ces commutateurs présentent deux problèmes sérieux dans les réseaux optiques : la conversion opto-électronique (O/E) et électro-optique (E/O) ajoute un délai additionnel au signal et la commutation électrique est lente comparée à la vitesse de propagation d'une fibre optique [1, 10, 26].

Les commutateurs optiques sont des dispositifs où les données sont maintenues sous forme optique. Le contrôle est toujours effectué électroniquement. Leur rôle principal est de fournir des longueurs d'onde à l'intérieur de la matrice optique ainsi qu'à protéger la fibre principale en cas de panne. Il existe plusieurs technologies utilisées dans leur fabrication, parmi lesquelles nous soulignons les commutateurs mécaniques, les commutateurs thermo-optiques et les commutateurs électro-optiques [16]. Dans les commutateurs mécaniques, la commutation est effectuée par l'intermédiaire de moyens mécaniques. Certains modèles utilisent des miroirs pour établir les longueurs d'onde. Ce type de commutateur est relativement économique, toutefois la vitesse de commutation est de l'ordre de la milliseconde. Les commutateurs électro-optiques utilisent un coupleur directionnel pour établir les canaux optiques. Dans ce cas, la fonction du coupleur est de varier l'indice de réfraction. Ces types de commutateurs opèrent à haute vitesse, de l'ordre de la nanoseconde. Finalement, les commutateurs thermo-optiques utilisent la chaleur pour faire varier l'indice de réfraction de la lumière et ainsi établir les connexions. Comme l'effet thermo-optique prend un certain temps, ces commutateurs opèrent à des vitesses de l'ordre de la milliseconde.

### 1.2.1 Commutation optique

Il existe deux paradigmes de commutation optique couramment utilisés :

1. La *commutation optique des circuits* (OCS) crée un canal optique (*lightpath*) qui va de l'origine jusqu'à la destination en utilisant une des longueurs d'onde disponibles entre chacun des commutateurs intermédiaires. De cette manière, une fois que le circuit est établi, les données y sont envoyées de manière totalement optique.

2. La *commutation optique des paquets* (OPS) traite chaque paquet tant dans le domaine optique que dans le domaine électronique. L'en-tête subit des conversions opto-électroniques à chaque fois qu'il arrive à un commutateur, tandis que la charge utile du paquet reste dans le domaine optique en utilisant pour cette fin la mémoire optique dans chaque commutateur avant de continuer son chemin vers sa destination. [4]

Toutefois, l'opération de lecture de données en OPS pourrait causer l'apparition d'un goulot d'étranglement dans les commutateurs. Les commutateurs opto-électroniques, nécessaires pour le traitement des paquets, ne peuvent pas atteindre les vitesses de traitement de la même manière que les réseaux optiques. Pour mettre en perspective cette situation, Ramasuni [16] donne un exemple en utilisant un paquet IP avec une taille de 53 octets. Un commutateur dans un réseau à 100 mégabit/s aura besoin de 4.24 $\mu$s pour le traiter. À 10 gigabit/s le même paquet sera traité dans 42.4 ns. Le design d'un commutateur assez rapide capable de traiter les paquets à la même vitesse que les composantes optiques pose encore des obstacles à surmonter à cause des limitations existantes dans les domaines physique et électronique.

Cette situation pourrait nous amener à choisir la commutation OCS, où les données sont traitées dans le domaine optique. Pourtant, quelques désavantages comme le gaspillage de la bande passante lors de périodes d'inactivité sur le canal ont motivé la recherche de nouvelles techniques de commutation pour les réseaux optiques. L'architecture des réseaux à commutation de rafales optiques (OBS) combine les avantages de la commutation optique de circuits et la commutation optique de paquets. Il s'agit d'une forme intermédiaire entre ces deux paradigmes. OBS a suscité une grande attention dans le milieu universitaire et industriel. Dans les sections suivantes, nous présentons un aperçu de cette technologie.

## 1.3 Architecture du réseau

La technique OBS évite deux principaux problèmes : le besoin de commuter les paquets à une vitesse élevée, au-delà des capacités de commutation actuels, et le stockage optique des données.

Il n'existe pas une définition précise d'un réseau OBS, toutefois la plupart des auteurs s'entendent sur les caractéristiques suivantes [4, 7] :

- Granularité ente OCS et OPS.
- Paquets de plusieurs tailles appelés rafales.
- Séparation entre l'information de contrôle et les données (charge utile).
- Opération asynchrone.
- La commutation de rafales ne requiert pas de stockage temporaire.
- Les ressources se réservent sans avoir besoin d'acquittement.

L'idée de base en OBS est l'envoi de plusieurs paquets dans une seule rafale, ou *burst*, sans qu'ils subissent la conversion optoélectronique à chaque commutateur

dans leur chemin. Le réseau est formé de commutateurs de frontière à l'entrée ou à la sortie et de commutateurs internes reliés ensemble par des liens de fibre optique [4]. La figure 1.3 illustre un exemple d'un réseau OBS. Aux commutateurs de frontière, les paquets arrivant du réseau d'accès seront récoltés et assemblés dans une rafale et par la suite celle-ci est envoyée sur le réseau. Une fois que la rafale arrive au commutateur de sortie, le processus de désassemblage a lieu, livrant les paquets à leur destination finale.

Puisque plusieurs paquets sont assemblés dans une unité plus grande, il y a plus de données par en-tête que dans les réseaux de paquet traditionnels. Par conséquent, un débit plus élevé est obtenu avec le même taux de traitement d'en-tête. En plus, aucun élément de stockage temporaire n'est nécessaire. Les rafales peuvent être stockés sous forme électronique à l'entrée du réseau au lieu de le faire à chaque noeud pendant que l'en-tête correspondant est traité. Par conséquent les commutateurs n'ont pas besoin de mémoire tampon optique (ou fibres de délai).

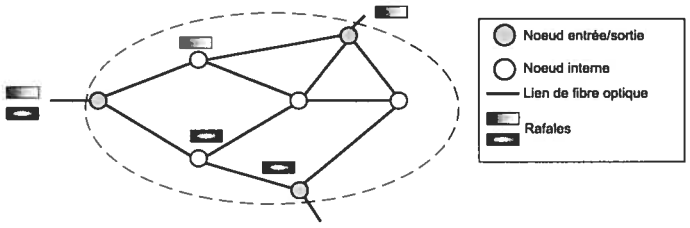



FIG. 1.3 – Réseau OBS

### 1.3.1 En-tête de contrôle

L'information de contrôle de la rafale comme la destination, le nombre de paquets qui la forment ainsi que la taille de la rafale, n'est pas incluse dans la rafale elle-même mais dans une unité séparée appelée *l'en-tête de contrôle*. Cette unité est envoyée à

l'avance, afin d'essayer d'établir un canal entre chaque commutateur qui sera utilisé par la rafale pour atteindre sa destination. Le temps écoulé entre l'expédition de l'en-tête de contrôle et la rafale correspondante s'appelle *décalage* (figure 1.4).

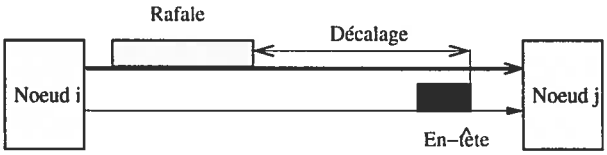



FIG. 1.4 – Décalage

### 1.3.2 Liens et canaux

Le lien qui joint deux commutateurs peut comporter plusieurs longueurs d'onde ou canaux. L'architecture de réseau définit deux ensembles de canaux [27]. Le premier ensemble est utilisé par les en-têtes de contrôle qui subiront une conversion opto-électronique à chaque commutateur, comme en OPS. Le deuxième ensemble de canaux est utilisé pour l'envoi des rafales. À la différence de l'en-tête de contrôle, les rafales traverseront le réseau en utilisant un chemin totalement optique, sans avoir besoin ni de stockage temporaire ni d'aucun traitement aux commutateurs intermédiaires.

### 1.3.3 Assemblage de rafales

C'est au commutateur d'entrée que des paquets provenant de plusieurs sources sont assemblés en rafales. Il existe une grande variété de critères utilisés pour inclure un paquet dans une rafale, dont le type de paquet, la priorité, la destination, etc. [29]. Selon la méthode d'assemblage, plusieurs files d'attente sont installées pour conserver les paquets pendant que la rafale est créée. À titre d'exemple, la figure

1.5 donne une illustration de la création d'une nouvelle rafale. Du côté gauche de l'unité d'assemblage se trouvent des paquets générés par les utilisateurs. Définissons-les comme *les sources.* Une fois qu'un seuil est atteint soit par période, par taille, ou par une combinaison de ces deux facteurs, les paquets correspondants sont assemblés dans la rafale qui est ensuite transmise.

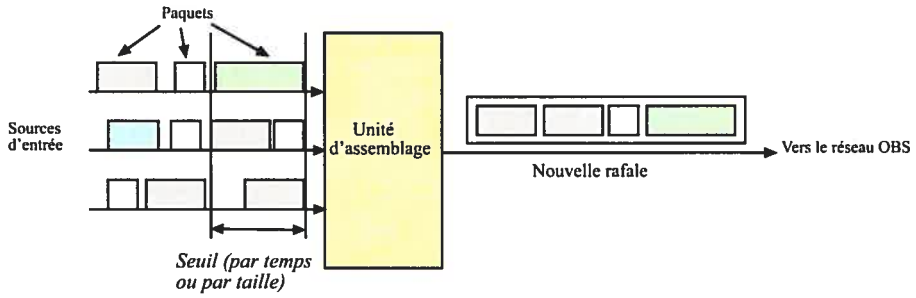



FIG. 1.5 – Assemblage de rafales

### 1.3.4 Commutateur OBS

Les réseaux OBS utilisent un commutateur spécialement conçu pour le traitement des rafales et des en-têtes de contrôle. La figure 1.6 présente un diagramme d'un commutateur, dans lequel nous illustrons le contrôleur et l'unité de commutation optique. Le canal de contrôle, par lequel voyagent les en-têtes de contrôle, est relié au contrôleur. C'est ici que prend place la conversion opto-électronique. Nous y trouvons également le mécanisme de réservation, qui réalise plusieurs tâches. D'abord, il détermine le canal de sortie que devra prendre la rafale ainsi que le paquet de contrôle. Ensuite, il se charge de mettre à jour la valeur du décalage et, finalement, il crée et maintient la table de routage du commutateur. Les canaux de données sont reliés directement à la *matrice optique de commutation* (OXC) [27]. Lorsqu'elles traversent le commutateur, les rafales restent dans le domaine optique.

Quelques auteurs suggèrent l'utilisation de *fibres de délai* (FDL) comme mémoire tampon optique pour réduire la probabilité de perte de rafales. Par contre, un des objectifs principaux dans la conception des réseaux OBS est d'avoir un réseau sans mémoire tampon, où les données voyagent à haute vitesse. La transmission de données sans mémoire tampon est importante puisque son utilisation requiert la conversion opto-électronique, ce qui ralentit la transmission de données [1]. Dans notre étude, nous ne considérons pas son utilisation.

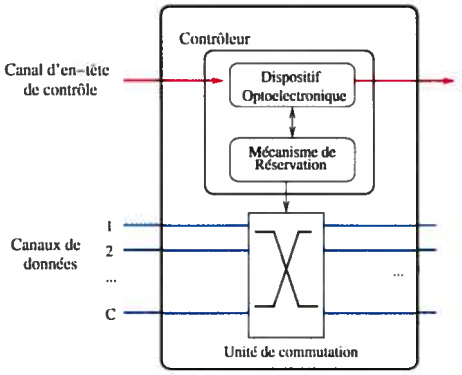



FIG. 1.6 – Commutateur OBS

### 1.3.5 Mécanisme d'envoi de rafales

Dans les réseaux OBS, les connexions entre un port d'entrée et un port de sortie d'un commutateur sont mis en place juste avant l'arrivée de la rafale. Ceci est possible grâce à un système de réservation mis en place à chaque commutateur. Dans ce but, la fonction de l'en-tête de contrôle est de réserver un canal à chaque commutateur pour la rafale correspondante. Le contrôleur lit l'information contenue dans l'en-tête et par la suite il exécute l'algorithme de réservation de canaux. Si la réservation réussit, l'en-tête continue au prochain commutateur qui se trouve sur sa route, en utilisant le canal consacré aux en-têtes de contrôle, alors que le contrôleur garde l'information

pour rendre le canal disponible au moment où il sera nécessaire. Lorsque la rafale arrive, elle traverse le commutateur à travers le canal réservé sans être retardée. Si aucun canal n'est disponible pour le temps demandé par la réservation, le contrôleur n'admet pas la rafale, qui sera perdue à son arrivée, ainsi que l'en-tête correspondant.

## 1.4 Contexte

Le contexte dans lequel nous allons appliquer l'algorithme de réservation de canaux dans les chapitres à venir est le suivant : nous cherchons l'implantation d'un mécanisme de réservation dans les commutateurs optiques ayant deux avantages essentiels : un faible temps de traitement au contrôleur et une faible probabilité de perte de rafales.

# Chapitre 2

# Modèles de réservation dans les commutateurs

Le but de ce chapitre est d'expliquer les raisons pour lesquelles un nouvel algorithme d'ordonnancement de rafales pour les réseaux OBS est justifiable. Nous allons commencer par discuter le rôle prépondérant que joue le décalage dans la conception des algorithmes de réservation. Ensuite, nous allons donner un aperçu des mécanismes de réservation dans les commutateurs optiques, en particulier les modèles Horizon et JET, couramment étudiés dans la littérature. La section 2.2 propose des idées originales en vue d'implanter un algorithme de réservation efficace pour les commutateurs optiques dans les réseaux OBS. Finalement, nous allons présenter d'autres approches pour l'ordonnancement de rafales basé sur la discrétisation.

## 2.1 Algorithmes de réservation

Rappelons que le modèle de réseau OBS propose une séparation entre l'information de contrôle et la charge utile. Le décalage, qui est le temps écoulé entre

l'expédition de l'en-tête de contrôle et la rafale, joue un rôle important car il permet aux commutateurs de faire la conversion opto-électronique et les calculs pour l'allocation de la rafale qui le précède.

Lorsque l'en-tête avec le décalage $\delta$ arrive au commutateur, le contrôleur prend un temps non négligeable pour faire la conversion opto-électronique et pour exécuter l'algorithme de réservation. Définissons $\tau$ comme la somme du temps requis pour la conversion opto-électronique, la lecture de l'information dans l'en-tête et le temps d'exécution de l'algorithme de réservation. Avant de continuer son chemin vers le prochain commutateur, l'en-tête aura un décalage réduit égal à $\delta - \tau$ (voir la figure 2.1).

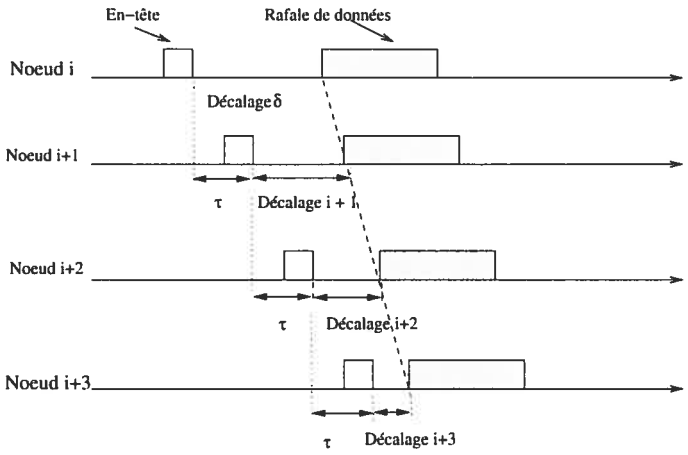



FIG. 2.1 – Mise à jour du décalage

Si $\tau > \delta$, la rafale arrivera avant qu'un des canaux ne soit disponible et par conséquent elle sera perdue. Cet événement dénoté comme une *arrivée précoce*, est fortement indésirable. Pour les éviter, il faut prévoir un décalage maximum dès l'entrée dans le réseau pour s'assurer que la rafale ne rattrapera pas l'en-tête avant l'arrivée à destination. Supposant que la plus longue route comporte $H$ commutateurs, un décalage initial ayant une valeur de $H\tau$ est suffisant pour assurer qu'aucune

arrivée précoce ne se produise. Notons que lorsque la valeur de $H$ dépend du nombre de commutateurs de transit sur le chemin, la valeur de $\tau$ dépend du temps de traitement de l'algorithme d'allocation dans les commutateurs. Ce temps de traitement varie selon l'algorithme choisi. Plusieurs algorithmes de réservation on été proposés pour les réseaux OBS. De façon générale nous pouvons les classifier en deux groupes [7] :

1. les mécanismes utilisant le principe *just-in-time* (JIT) et
2. les mécanismes utilisant le principe *just-enough-time* (JET).

La principale différence entre eux est la façon d'indiquer le début et la fin de la rafale ainsi que l'instant où la réservation du canal commence. Sous l'approche JIT, l'en-tête de contrôle contient l'information concernant l'arrivée de la rafale. Dans le commutateur, le contrôleur garde une liste indiquant l'état des canaux (libres ou occupés) pour chaque port de sortie. Lorsque la demande de réservation arrive au commutateur, si un canal est disponible, la rafale sera allouée sur ce canal à partir de l'arrivée de la demande de réservation. Autrement, la rafale sera perdue. Une amélioration à ce mécanisme consiste à réserver le canal pour une durée limitée. Pour ce faire, la taille de la rafale est incluse dans les données à transporter par l'en-tête de contrôle. En effet, l'algorithme Horizon [22] utilise ce principe. Dans ce cas, le canal sera libéré une fois que la transmission de la rafale se termine. La liste d'état de canaux contient l'information sur l'instant où chaque canal sera libéré (aussi appelé l'horizon dans [22]). Même si le temps de traitement de l'algorithme Horizon est faible, ce dernier gaspille de la bande passante, puisqu'aucune rafale ne peut être allouée dans les intervalles libres entre les réservations.

Les mécanismes JET [14] considèrent l'instant d'arrivée et de départ de chaque rafale. Ceci leur permet d'atteindre une meilleure efficacité que celle de l'algorithme Horizon car il est possible d'utiliser les intervalles libres entre deux rafales afin d'allouer une nouvelle demande de réservation. Dans ce cas, le contrôleur maintient une liste contenant le temps d'arrivée et le temps de départ de *toutes* les rafales qui ont fait une réservation aux ports de sortie du commutateur. Comme la recherche d'un canal libre aux ports de sortie s'effectue sur cette liste, l'algorithme JET présente un temps de traitement élevé.

## Notation

Soit $C$ le nombre de canaux disponibles dans chaque commutateur $p$. Le contrôleur associé à ce commutateur contient un vecteur indiquant l'état de la liste de réservations. Pour chaque canal $i = 1, \ldots, C$ soit $\mathrm{n}(i)$ le nombre de rafales qui ont fait une réservation sur ce canal. Supposons que la $n^{\mathrm{e}}$ rafale demande une réservation à l'instant dénotée par $t_n$ et que la taille de la rafale est de $b_n$. Chaque réservation prend un intervalle de transmission dénoté par $l$. Nous allons utiliser $s_l$ et $e_l$ pour dénoter le début et la fin de l'intervalle $l$ respectivement. Nous supposons que l'index $l$ trie les réservations en ordre croissant, c'est-à-dire, pour tout $l = 1, \ldots, \mathrm{n}(i) - 1, e_l(i) \leq s_{l+1}(i)$.

### 2.1.1 Procédure de réservation avec Horizon

Dans le cas de l'algorithme Horizon, la liste de réservation du canal $i$ contient la fin de l'intervalle de la rafale qui a fait la dernière réservation. Horizon alloue la nouvelle rafale dans le canal libre de réservations offrant le plus petit gaspillage de bande passante.

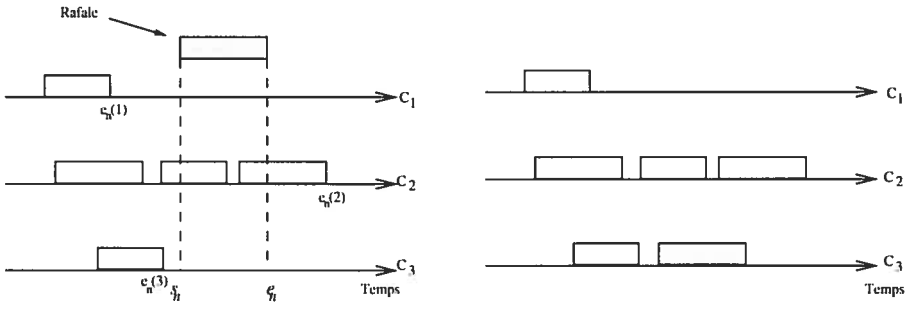
En termes mathématiques, définissons $C_n$ comme l'ensemble de canaux où la demande de réservation faite par la $n^e$ rafale est susceptible d'être allouée. De même, définissons $c_n$ comme le canal qui recevra cette réservation. Alors nous avons :

$$C_n = \{i : e_{\mathrm{n}(i)}(i) \leq s_n\} \tag{2.1}$$

$$c_n = \arg\min_{i \in C_n} \left(s_n - e_{\mathrm{n}(i)}(i)\right) \tag{2.2}$$

Si aucun canal n'est disponible, la rafale est perdue, autrement, $\mathrm{n}(i) = \mathrm{n}(i) + 1$ et $e_{\mathrm{n}(i)}(i) = e_n$

Nous illustrons dans la figure 2.2 un exemple de l'algorithme Horizon. La rafale en haut de la figure a besoin d'un canal libre au port de sortie. L'ensemble des canaux candidats, $C_n$, est formé par le canal $C_1$ et $C_3$. Le canal qui recevra la réservation, $c_n$, est $C_3$ car le vide qui sera créé entre le temps d'arrivée de la nouvelle rafale, $s_n$, et le départ de la rafale qui est déjà sur ce canal, $e_{\mathrm{n}}(3)$, est plus petit que celui sur le canal $C_1$ (voir figs. 2.2a et 2.2b).




a) Demande de réservation b) Réservation allouée.

FIG. 2.2 – Modèle de réservation avec Horizon

Horizon exige simplement une recherche de l'horizon c'est-à-dire le temps de

départ de la dernière réservation pour chaque canal. Il a donc besoin d'un temps de traitement de l'ordre du nombre de canaux, dans le pire des cas, ce qui lui permet d'avoir un temps constant et non aléatoire. Cependant, Horizon manque d'efficacité à cause de son mécanisme d'allocation de rafales qui ne permet pas la recherche des intervalles libres entre réservations.

### 2.1.2 Procédure de réservation avec JET

Dans le cas de l'algorithme JET, l'état de la liste de réservation du canal $i$ contient le debut $s_l(i)$ ainsi que la fin $e_l(i)$ des intervalles occupés par des rafales alloués dans le commutateur.

Soit $s_{n(i)+1} = \infty$. La recherche dans la liste détermine les canaux candidats, $C_n$, ayant des intervalles libres pour allouer la nouvelle rafale. Par la suite, le canal qui produit le plus petit intervalle libre, $c_n$, sera choisi. En termes mathématiques, cette situation s'exprime par :

$$l^*(i) = \arg\max_l (e_l(i) < s_n)\, i = 1, \ldots, C \tag{2.3}$$

$$C_n = \{i\colon s_{l^*+1}(i) > e_n\} \tag{2.4}$$

$$c_n = \arg\min_{i \in C_n} (s_n - e_{l^*}(i)) \tag{2.5}$$

En ce qui concerne la dernière équation, il est aussi possible de choisir le canal de la façon suivante :

$$c_n = \arg\min_{i \in C_n} (s_{l^*+1}(i) - e_n) \tag{2.6}$$

Pour déterminer les canaux candidats, la recherche en JET s'effectue en deux dimensions, c'est-à-dire entre le point d'arrivée et le point de départ des réservations existantes. Cette recherche est effectuée sur une liste ayant un nombre *aléatoire* d'éléments, de sorte que l'algorithme prend un temps aléatoire même lorsque le nombre de canaux est connu à l'avance. Un exemple de réservation avec l'algorithme JET est illustré dans la figure 2.3. En haut de la figure, nous avons la rafale ayant besoin d'un canal au port de sortie du commutateur. Chaque canal contient une liste indiquant le début $s_l(i)$, et la fin $e_l(i)$, des intervalles occupés par les rafales déjà sur place. Afin de déterminer l'ensemble des canaux candidats nous utilisons l'équation (2.4) ce qui nous donne les canaux $C_2$ et $C_3$. En employant l'équation (2.5), le canal qui produit l'espace libre le plus petit, $c_n$, est $C_3$ (fig. 2.3b).

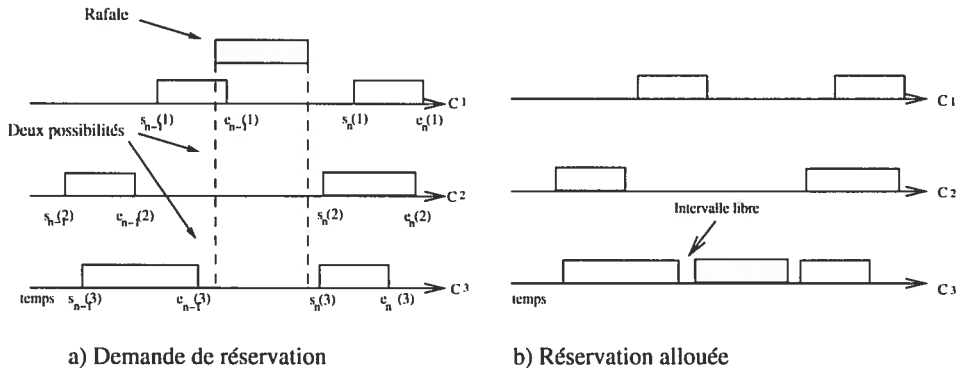



FIG. 2.3 – Modèle de réservation avec JET

L'algorithme JET produit une probabilité de perte de rafales plus faible grâce à son mécanisme d'utilisation des intervalles entre réservations, mais cet avantage apparent est surpassé par sa complexité informatique. Un algorithme ayant une complexité informatique élevée implique un temps de calcul plus élevé. Récemment il a été démontré que le temps de calcul de JET augmente selon la charge par lien [23]. Le fait que ce temps de calcul n'est pas constant mais dépend de la charge du réseau

rend difficile, sinon impossible, le calcul d'un décalage suffisant à l'entrée du réseau. De plus, un temps de calcul élevé entraîne des effets indirects sur le contrôleur, tel que le traitement des demandes de réservation simultanées lorsque plus d'un en-tête de contrôle arrive au cours du traitement de la demande. Même si l'utilisation de fibres de délai est possible pour stocker de façon temporaire les en-têtes de contrôle, les réseaux OBS sont mieux conçus sans l'utilisation de files d'attente pour éviter, entre autres, une variation aléatoire des décalages en aval. Un algorithme ayant un temps de traitement faible est donc essentiel pour réduire la probabilité de traiter deux ou plusieurs demandes en même temps.

En comparant les deux algorithmes il est évident que :

- La différence principale entre Horizon et JET en termes de temps de traitement CPU est due à la détermination des canaux candidats, d'après (2.1) et (2.4), Comme nous le montrerons, il s'avère que cette opération qui exige un nombre aléatoire de comparaisons peut être plusieurs fois plus coûteuse que le reste de l'algorithme JET.
- Le nombre de comparaisons entre (2.1)-(2.2) et entre (2.4)-(2.5) est dans le pire des cas dans l'ordre du nombre de canaux.

À cause de la dégradation du temps de calcul de l'algorithme JET, quelques auteurs [25] préfèrent le mécanisme de l'algorithme Horizon par rapport à celui de JET. Horizon a un temps de traitement qui est dans le pire cas de l'ordre du nombre de canaux, ce qui lui permet d'avoir un temps constant et non aléatoire, alors que l'algorithme JET a un temps de traitement aléatoire, dépendant de l'état des réservations déjà en place.

Nous proposons S-JET, un algorithme de réservation efficace, ayant le meilleur des deux mondes : la faible probabilité de perte de l'algorithme JET, et le temps de traitement constant de l'algorithme Horizon. S-JET[1] utilise les intervalles entre les réservations mais il présente un temps de traitement qui est de l'ordre du nombre de canaux. Le reste du chapitre est consacré à la présentation de cet algorithme.

## 2.2 Procédure de réservation dans S-JET

La méthode présentée ici fonctionne selon une conception de réseau où :

- les rafales arrivant dans n'importe quel commutateur ont une taille *maximale* $\bar{b} < \infty$ mesurée selon les unités de temps qu'elles prennent pour la transmission,
- le décalage possible maximal des en-têtes entrants $\bar{\delta}$ est connu à l'avance.

La première condition exige que les noeuds d'entrée envoient des rafales à chaque fois que l'information multiplexée atteint un seuil de taille spécifique. En utilisant le modèle de Vázquez-Abad *et al.* [23] comme repère, le décalage possible maximal est indiqué par $\bar{\delta} = H\tau$, où $H$ est le nombre maximum de sauts dans une route, et $\tau$ est le délai total au contrôleur, la conversion opto-électronique plus le temps de traitement de l'algorithme. Comme mentionné auparavant, ce modèle établit un premier décalage qui est tout juste assez grand pour éviter des arrivées précoces, ce qui entraînerait la perte des données. D'autres attributions de décalages sont possibles dans S-JET, tant que le décalage maximal est connu et fixé à l'avance.

À partir de la description mathématique des algorithmes dans les sections 2.1.1 et 2.1.2, il devient évident qu'un algorithme efficace doit viser à atteindre une phase

[1] Slotted-JET

de recherche dans un temps de traitement négligeable. L'idée derrière S-JET est la discrétisation de la rafale et du temps de réservation. Nous définissons une case[2] comme un intervalle du temps de taille constante $\Gamma$. À partir d'ici, les équations (2.3) et (2.4) s'implémentent à l'aide d'opérations booléennes, réduisant ainsi le temps d'exécution de l'algorithme.

Soit $\mathbf{x}(n)$ un vecteur qui représente les cases nécessaires pour allouer la demande de réservation de la $n^{\text{e}}$ rafale de taille $b_n \leq \bar{b}$. En d'autres termes, $x_j(n) = 1$ si la rafale a besoin que la $j^{\text{e}}$ case soit réservée, autrement $x_j(n) = 0$. Puisque la liste de réservation est dynamique, nous utilisons le vecteur $\mathbf{r}(n)$ pour indiquer l'état de la liste de réservations pour tous les canaux. Ce vecteur est formé de cases de taille $\Gamma$ et l'horizon est défini par $\mathcal{T}$.

Nous utilisons $\mathbf{y}$ pour nommer un vecteur qui résulte des opérations booléennes intermédiaires. Les éléments de ce vecteur sont définis pour chaque canal $i = 1, \ldots, C$, comme $y^j \; \forall \, j = 1, \ldots, \lceil \mathcal{T}/\Gamma \rceil$. Pour tous le vecteurs utilisés dans cette section, les éléments de chaque vecteur peuvent avoir une valeur de 0 ou 1. Pour simplifier l'explication du principe de S-JET, nous supposons que tous les vecteurs sont de longueur égale.

En considérant n'importe quel commutateur dénoté par $p$, S-JET consiste à réaliser les étapes suivantes :

1. Mettre à zéro le vecteur de réservation pour chaque canal.

$$r_i^j(n) = 0 \quad \forall \, j = 1, \ldots, \lceil \mathcal{T}/\Gamma \rceil \tag{2.7}$$

[2] slot

où l'index $j$ répresente le numéro de case et $i$ le canal dans le port de sortie.

2. À l'arrivée d'un en-tête de contrôle pour la $n^{\text{e}}$ rafale, soit $x(n)$ la représentation binaire de la rafale et $t_n$ le début de la réservation pour cette rafale :

$$x_j(n) = \begin{cases} 1 & \text{si } t_n \leq (j-1)\,\Gamma \leq t_n + b_n \\ 0 & \text{autrement,} \end{cases} \tag{2.8}$$

pour $j = 1, \ldots \lceil \mathcal{T}/\Gamma \rceil$.

3. Pour l'arrivée quelconque d'une rafale $n > 0$ et pour chaque canal $i = 1, \ldots, C$, vérifier si la réservation est possible. Ceci est effectué en exécutant l'opération booléenne AND suivante :

$$\mathbf{y}_i = \mathbf{r}_i(n) \wedge \mathbf{x}(n) \tag{2.9}$$

si $y_i^j = 0, \forall j = 1, \ldots, \lceil \mathcal{T}/\Gamma \rceil$, alors la réservation est possible dans le canal $i$, et nous mettons $i$ dans l'ensemble de candidats $C_n$. Autrement, la réservation dans le canal $i$ n'est pas possible.

4. Si $C_n = \emptyset$ alors la rafale sera perdue.

5. Dans la liste de canaux candidats, $C_n$, nous devons trouver le canal qui produit la perte la plus petite, $c_n$, selon l'équation (2.5). Pour ce faire, nous déterminons d'abord la taille de l'espace vide pour chaque canal dans $C_n$. Soit $j_n^* = \min\{j \colon x^j(n) = 1\}$ la première case occupée par la représentation discrète de la rafale $n$ et $\mathbf{Z}_i$ le nombre binaire qui résulte en tronquant le vecteur $\mathbf{r}_i(n)$ à la case $j_n^* - 1, i \in C_n$,

$$\mathbf{Z}_i = r_i^j(n), \quad j = 1, \ldots, j_n^* - 1.$$

Alors on peut montrer directement que le canal $c_n$ ayant la perte la plus petite est celui qui a la représentation en nombre entier la plus élevée, de sorte que

$$c_n = \arg \max_{1<i<C} \text{Int}(\mathbf{Z}_i) \tag{2.10}$$

6. Finalement, le vecteur de réservation est mis à jour pour ce canal en effectuant une opération booléenne OR entre le vecteur $\mathbf{x}$ et le vecteur de réservation précédent $\mathbf{r}$ :

$$\mathbf{r}_{c_n}(n+1) = \mathbf{x} \vee \mathbf{r}_{c_n}(n), \tag{2.11}$$

tandis que tous les autres canaux garderont le vecteur précédent de réservation :

$$\mathbf{r}_i(n+1) = \mathbf{r}_i(n), \forall i \neq c_n \tag{2.12}$$

La procédure décrite ci-dessus exige en effet des opérations booléennes mineures afin de déterminer les canaux candidats $C_n$ à chaque commutateur $p$. Ces opérations sont indépendantes des conditions du trafic et le temps de traitement est proportionnel au nombre de canaux $C$. Comme nous le montrerons au chapitre 4 section 4.3, il s'avère que la complexité informatique de l'algorithme S-JET est beaucoup moins lourde que celle de l'algorithme JET.

## 2.3 Autres approches

La méthode de recherche de l'algorithme S-JET est une approche originale utilisée pour choisir le meilleur canal pour allouer la rafale. Ramamirtham *et al* [15] proposent un mécanisme appelé «Time Sliced Optical Burst Switching»(TSOBS) visant l'élimination des convertisseurs de longueur d'onde dans les commutateurs et réduisant en même temps la taille de la mémoire tampon optique ou fibres de délai.

Dans leur implémentation, les canaux ou longueurs d'onde, sont divisés en trames, et chaque trame est à la fois subdivisée en plusieurs *timeslots* (cases). Dans les commutateurs intermédiaires, les fibres de délai sont essentielles car celles-ci sont utilisées par le mécanisme de réservation. La procédure pour allouer une rafale construit d'abord un arbre des plus courts chemins. Ensuite, elle cherche sur cet arbre une séquence de fibres de délai parmi lesquelles le timeslot sera commuté au port de sortie. L'algorithme utilise un tableau bidimensionnel $s[i,t]$, pour représenter les fibres de délai et les timeslots par trame où $s[i,t] = 1$ si la fibre de délai $i$ est occupée à l'instant $t$.

Même si les modèles TSOBS et S-JET emploient la discrétisation du temps et des tableaux bidimensionnels pour indiquer l'état des canaux dans les ports de sortie, les deux approches fonctionnent de façon différente. L'algorithme S-JET utilise des opérations booléennes pour déterminer le meilleur canal, augmentant en même temps l'utilisation de la bande passante. Par contre, dans la méthode de recherche utilisée avec TSOBS, une nouvelle rafale est allouée au timeslot sans considérer le gaspillage de bande passante qui sera produit une fois la nouvelle rafale allouée. Par conséquent, un grand nombre de petits intervalles libres sont créés où seulement des rafales

futures de petite taille pourront être allouées.

Finalement, notre approche n'a pas besoin de fibres de délai, lesquelles selon le même auteur [15], sont très coûteuses à utiliser. Dans le chapitre suivant, nous allons décrire l'implantation de l'algorithme S-JET.

# Chapitre 3

# Implantation de S-JET

L'algorithme S-JET a été conçu de façon similaire à JET pour profiter des intervalles libres entre les rafales déjà sur les canaux. Cependant, le mécanisme de réservation de l'algorithme S-JET présente un temps de traitement au contrôleur plus avantageux que celui de l'algorithme JET. Il y a deux façons de mettre en application S-JET : soit directement sur une puce (matériel) ou par logiciel. La mise en oeuvre matérielle permet un résultat optimal en termes de temps de traitement dans le contrôleur grâce à l'utilisation des portes logiques lors de l'exécution des opérations booléennes. Le temps de CPU requis pour ces opérations est négligeable. Cependant, l'implantation directe sur une puce entraînerait des coûts additionnels, par conséquent il est préférable d'utiliser des simulations au lieu d'expériences réelles. Dans ce chapitre, nous expliquons la mise en oeuvre logicielle de S-JET.

À partir de la description de base de l'algorithme dans la section 2.2, il est évident que le codage sera étroitement lié à la représentation du nombre entier qui sera choisi pour les variables. Pour nos programmes, nous utilisons le simulateur T-Sim[1],

[1] Voir Annexe A

développé avec le logiciel de simulation d'événement discret SSJ [9], écrit en Java.

## 3.1 Structure des données

Nous utilisons un vecteur bidimensionnel de $C$ lignes et $\mathcal{T}$ colonnes, pour représenter le vecteur $\mathbf{r}(n)$ de l'équation (2.7). Chaque élément du tableau est un nombre entier du type `long` (voir figure 3.1). Le vecteur $\mathbf{x}(n)$ en (2.8) est également représenté par un vecteur dans notre programme. Nous considérerons que chacun des 64 bits composant un nombre entier du type `long` représente une case de taille $\Gamma = \bar{b}/M$. D'autres types de codage peuvent être utilisés. Comme nous le mentionnerons plus tard, le temps de calcul est indépendant de la représentation choisie. Plus la valeur de $\Gamma$ est petite, plus les résultats obtenus sont exacts. Malheureusement, dans notre cas les restrictions imposées par le langage de programmation ne nous permettent pas d'utiliser une valeur de $\Gamma$ plus petite, donc plus précise. D'autres obstacles peuvent surgir, comme l'utilisation de la mémoire, qui pourrait être un facteur influent dans la mise en oeuvre matérielle de S-JET. Basés sur nos études préliminaires sur la discrétisation, et la limitation imposée par Java, nous avons choisi une représentation de 64 bits comme base pour nos simulations. Nous allons utiliser la variable $M$ pour faire référence à ce nombre de bits.

Soit $\sigma$ égal au nombre de bits pour représenter un nombre entier. Afin de représenter toutes les rafales avec la précision requise, nous imposons la condition

$$M \leq \sigma \tag{3.1}$$

Cette restriction est dérivée de Java où, afin d'exécuter des opérations booléennes entre deux variables, les deux variables doivent être des nombres entiers de type primitif (soit `byte`, `short`, `int` ou `long`) [6].

Pour faire une utilisation efficace de la mémoire RAM disponible, nous limitons l'horizon du vecteur de réservations, $\mathcal{T}$, à $\mathcal{T} = \bar{b} + \bar{\delta}$. Ce paramètre permettra à l'en-tête de contrôle de faire la demande de réservation sans risquer que sa demande soit rejetée pour dépassement de la limite de l'horizon. La manipulation de la liste sera faite de manière cyclique. Au temps d'initialisation, tous les bits du tableau sont mis à 0.

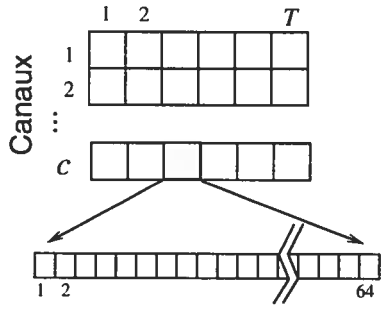



FIG. 3.1 – Structure des données, avec $\sigma = 64$

## 3.2 Procédure de réservation

L'équation (2.9) décrit une opération booléenne AND entre la variable $\mathbf{x}(n)$ et le vecteur de réservation $\mathbf{r}_i(n)$. Pour assigner une rafale à l'un des canaux nous illustrons cette opération à l'aide de la figure 3.2. En haut de la figure, nous observons la rafale ayant besoin d'un canal libre. La première étape consiste donc en la création du vecteur $\mathbf{x}(n)$. Les cases dans le vecteur $\mathbf{x}(n)$ correspondants à l'intervalle de temps nécessaire pour transmettre la rafale sont mises à 1 (fig. 3.2a). Le code employé pour

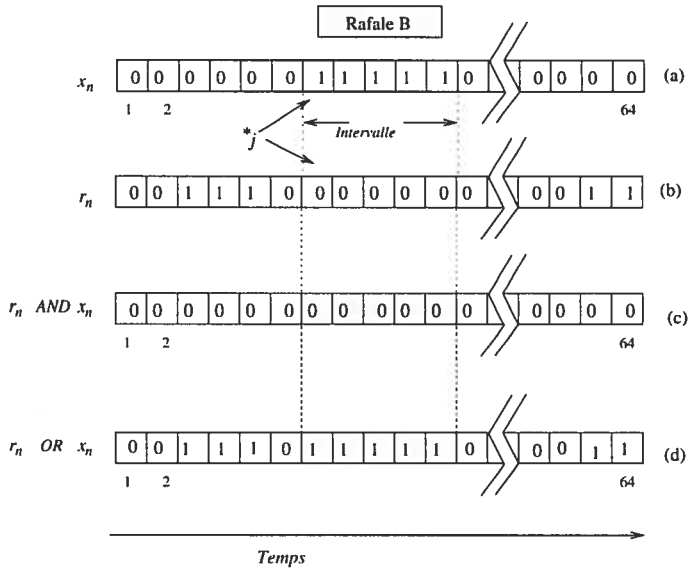



FIG. 3.2 – Schème de réservation avec S-JET

créer un tel masque se lit comme suit :

```
/**
  * Takes a slotted burst to create a long integer that will
  * represent the mask to be used in the AND operation.
  * In this case, the burst takes only one cell of the array
  *
  * @param numero        A long integer with all its bits set to 1.
  * @param desplaza      The first bit requested by the reservation
  * @param Sburst        The slotted burst
  */
private long toMask (long numero, long desplaza, Sburst b){
    long off = bits_per_integer_ - (desplaza + b.getNumberOfSlots());
    long mask = numero << off;
    mask = mask >>> (off + desplaza);
    mask = mask << desplaza;
    return mask;
}
```

L'état actuel de la liste de réservation $\mathbf{r}_i(n)$ apparaît à la figure 3.2b. Nous observons que les cases marquées avec 1 indiquent que la case est déjà réservée. L'étape suivante consiste à effectuer l'opération booléenne AND entre $\mathbf{r}_i(n)$ et $\mathbf{x}(n)$. Le résultat de cette opération est stocké dans le vecteur temporaire $\mathbf{y}$, illustré à la figure 3.2c. La

rafale peut occuper un canal seulement si $\mathbf{y} = 0$, selon l'équation (2.9).

Finalement, si la réservation est possible, il ne nous reste que la mise à jour du vecteur $\mathbf{r}_i(n)$ par une opération booléenne OR entre $\mathbf{r}_i(n)$ et $\mathbf{x}(n)$. Nous illustrons dans la figure 3.2d l'état final du vecteur $\mathbf{r}_i(n)$ après cette opération.

La demande d'une réservation pourrait arriver entre deux cellules consécutives du vecteur (voir la figure 3.3). Définissons $M_c$ comme le nombre de cases occupées par la réservation sur la cellule courante, et $M_{c+1}$ les cases occupées dans la cellule suivante. L'opération décrite ci-dessus sera effectuée deux fois, la première en utilisant les cases $M_c$ de la cellule $c$ et l'autre les cases $M_{c+1}$ de la cellule $c + 1$. Clairement $M_c + M_{c+1} \leq M \leq \sigma$.

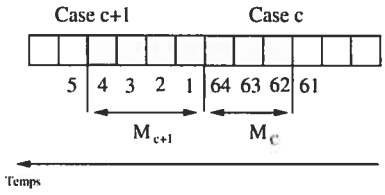



FIG. 3.3 – Une rafale qui occupe deux cellules consécutives

## 3.3 Une procédure rapide pour choisir le meilleur canal

Selon l'équation (2.9), $C_n(p)$ est l'ensemble de canaux où la réservation de la nouvelle rafale est possible. L'algorithme S-JET choisit un de ces canaux pour allouer la réservation. À l'aide de l'équation (2.10), S-JET choisit le canal $c_n(p)$ ayant le moins de cases qui seront laissées vides, ce que permet une utilisation plus efficace des ressources.

Rappelons que $j^*$ est la première case occupée par la rafale. Une exécution rapide de S-JET choisit $c_n(p)$ en transformant en nombre entier la représentation binaire équivalente des $\sigma$ cases précédant $j^*$ dans $\mathbf{r}_i(n)$ pour chaque canal en $C_n(p)$. Il s'agit d'une troncation de $\mathbf{r}$ utilisant seulement les premières $\sigma$ cases après la case $j^*$. Selon l'équation (2.10), le canal choisi est celui avec le nombre décimal ayant la valeur la plus élevée. Il faut souligner que cette valeur n'a aucune relation avec la taille de la rafale. L'implantation de ce mécanisme, qui utilise des opérateurs de décalage, nous permet la manipulation directe des bits dans les variables sans utiliser aucune opération arithmétique. Nous obtenons également la valeur décimale recherchée. Le code de ce procédé est le suivant :

```
private long findVoidValue(long currentCell,
                           long previousCell,
                           long offset) {
if (offset==0)
       return (previousCell);
else {
       long m1 = currentCell << (bits_per_integer_-offset);
       long m2 = previousCell >>> offset;
       long r =  m1|m2;
       if (r<0) r = r >>> 1;
       return (r);
     }
}
```

Nous illustrons à la figure 3.4 ce procédé. En bas de la figure, nous avons la rafale ayant besoin d'un canal au port de sortie. L'ensemble de canaux candidats, $C_n(p)$, est formé par $C_2$ et $C_3$. Nous supposons que la valeur de $\sigma$ est égale à 64. Si S-JET choisit le canal $C_3$, une case sera laissée libre. D'autre part si le canal $C_2$ est choisi, il en résultera un espace de 4 cases. Afin d'obtenir la valeur décimale $\mathbf{Z}$ de ces intervalles vides, l'algorithme S-JET considère les cases $j^* - 1$ à $j^* - 64$ pour remplir les $\sigma$ bits d'une variable de type `long` en Java. Le bit plus significatif de

cette variable est la case $j^* - 1$. La valeur décimale pour le canal $C_2$ est $\mathbf{Z}_2 = 2^1 + 2^2 + 2^3 + 2^7 + 2^{59} \approx 2\text{x}10^{59}$ et $\mathbf{Z}_3 = 2^1 + 2^2 + 2^3 + 2^5 + 2^6 + 2^7 + 2^{60} + 2^{61} + 2^{62} \approx 2.22 \times 10^{62}$. Puisque $\text{Int}(\mathbf{Z}_3) > \text{Int}(\mathbf{Z}_2)$, S-JET choisit le canal $C_3$.

Il est à noter que le plus petit espace pourrait également être trouvé simplement en évaluant la valeur du bit le plus significatif. Dans cet exemple, cela donnerait $\mathbf{Z}_2 = 2^{59}$ et $\mathbf{Z}_3 = 2^{62}$.

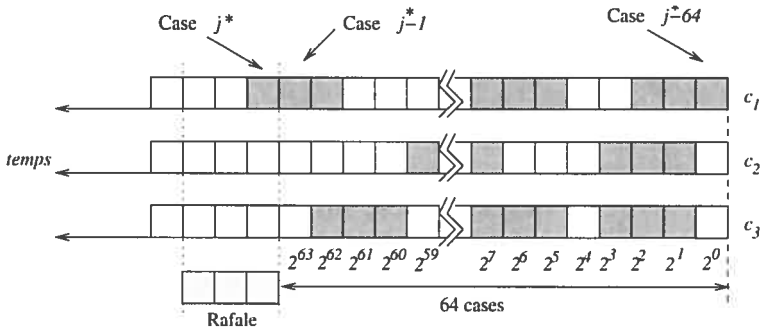



FIG. 3.4 – Sélection du canal avec le plus petit nombre de cases vides

## Blocage additionnel

La procédure pour choisir le meilleur canal considère seulement les premières $\sigma$ cases après $j^*$ pour créer le nombre entier, mais ceci ne nous limite pas à pousser plus loin nos explorations de cases, tels que $2\sigma$, $3\sigma$ et ainsi de suite. En définissant la méthode présentée dans la section 2.2 comme la version *théorique*, l'approche adoptée dans l'implantation logicielle correspond à une version *myope* de la version théorique (éq. 2.10). La version myope peut, dans certains cas, produire une source de blocage additionnelle. Cependant, dans d'autres situations, elle peut réduire le

blocage. Ceci est expliqué par les exemples dans les figures 3.5 et 3.6, où S-JET myope est comparé à la version théorique de S-JET.

Pour déterminer le meilleur canal la recherche myope regarde seulement les $\sigma$ cases dans chacun des canaux candidats. Ce champ de vision est égal à la taille de la rafale la plus grande qui peut être envoyée sur le réseau. S-JET théorique évalue la taille de chaque intervalle libre dans les canaux candidats. À partir de ces évaluations, S-JET choisit le canal $C_2$ car c'est ici que la perte est réduite au minimum. Si la recherche myope trouve deux ou plus canaux ayant la même valeur décimale, elle choisit le premier des canaux disponibles trouvés. Dans cet exemple, elle assigne le canal $C_1$ à la réservation entrante, marquée avec l'étiquette (1) dans la figure 3.5a. Quand la rafale avec l'étiquette (2) demande un canal, (fig. 3.5b), les deux versions de S-JET trouvent un canal libre. Ainsi les deux implantations placent correctement la rafale (2) dans le canal ayant le nombre le plus petit de cases vides. Dans cet exemple, le champ de vision de S-JET myope est suffisant pour compter le nombre total de cases entre réservations. La figure 3.5c illustre ceci. Le problème apparaît pour la réservation demandée par l'en-tête de contrôle de la rafale (3). S-JET théorique a un intervalle libre pour assigner la réservation, alors que la version myope ne l'a pas (figure 3.5d).

La situation inverse peut également se produire. Ceci est illustré dans la figure 3.6 où S-JET théorique bloque la réservation demandée par la rafale (3) tandis que la version myope l'accepte. Il y a d'abord une demande faite par la rafale (1). S-JET myope assigne la rafale sur le canal $C_1$ tandis que la version théorique le fait sur le canal $C_2$ (figure 3.6a). Quand la nouvelle demande de réservation de la rafale (2) arrive, les deux versions sont capables de l'admettre : la version myope va l'admettre

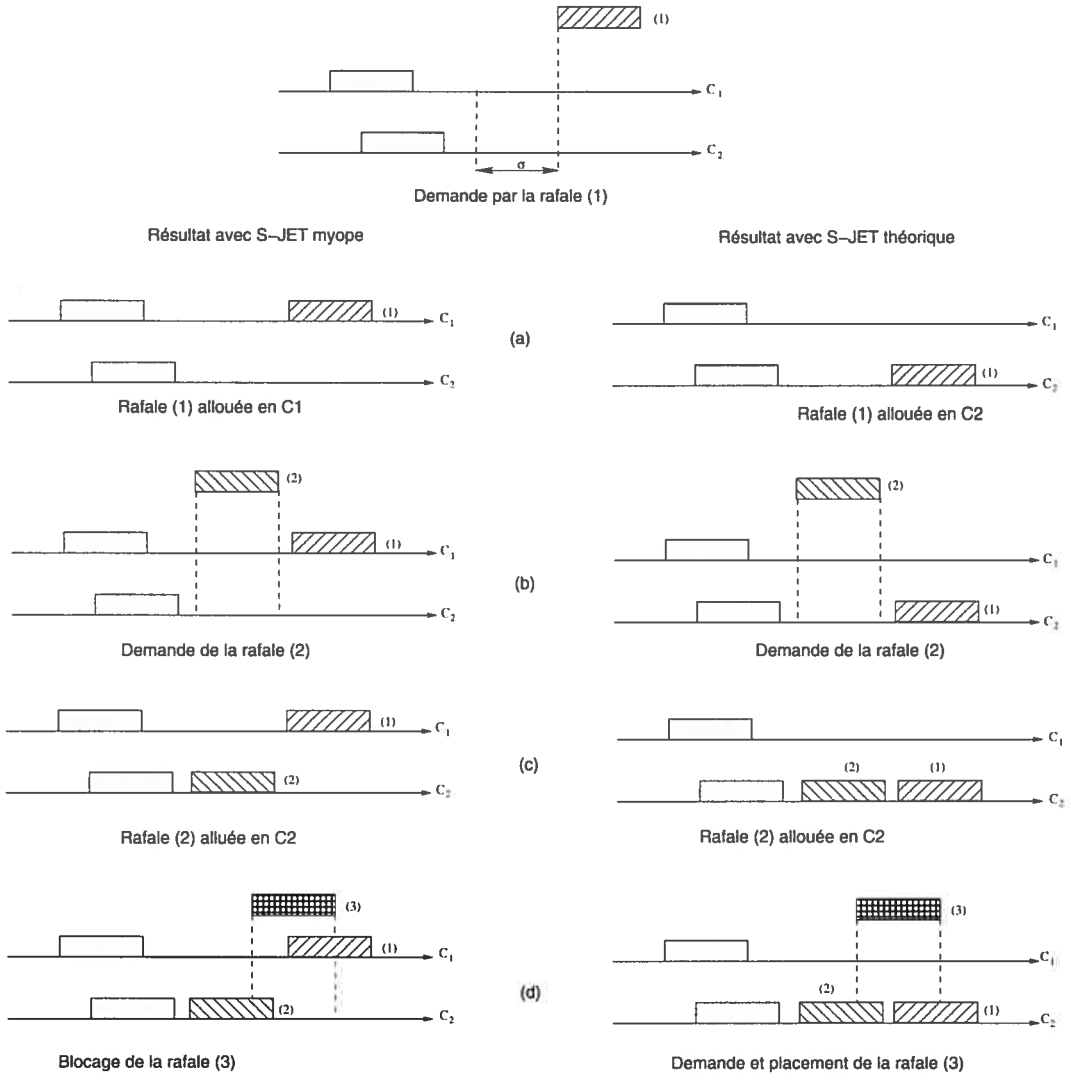



FIG. 3.5 – Blocage d'une réservation avec S-JET *myope*

sur le canal $C_2$ tandis que la version théorique le fera sur le canal $C_1$ (voir la figure 3.6c). Cependant, quand la réservation suivante arrive, la version myope de S-JET l'accepte tandis qu'elle est refusée par la version théorique de S-JET.

La proposition suivante va nous permettre de montrer les propriétés de la méthode S-JET. Nous allons supposer que le processus d'arrivée des rafales est un processus Poisson. Cette hypothèse ne peut pas être confirmée puisque l'arrivée des rafales

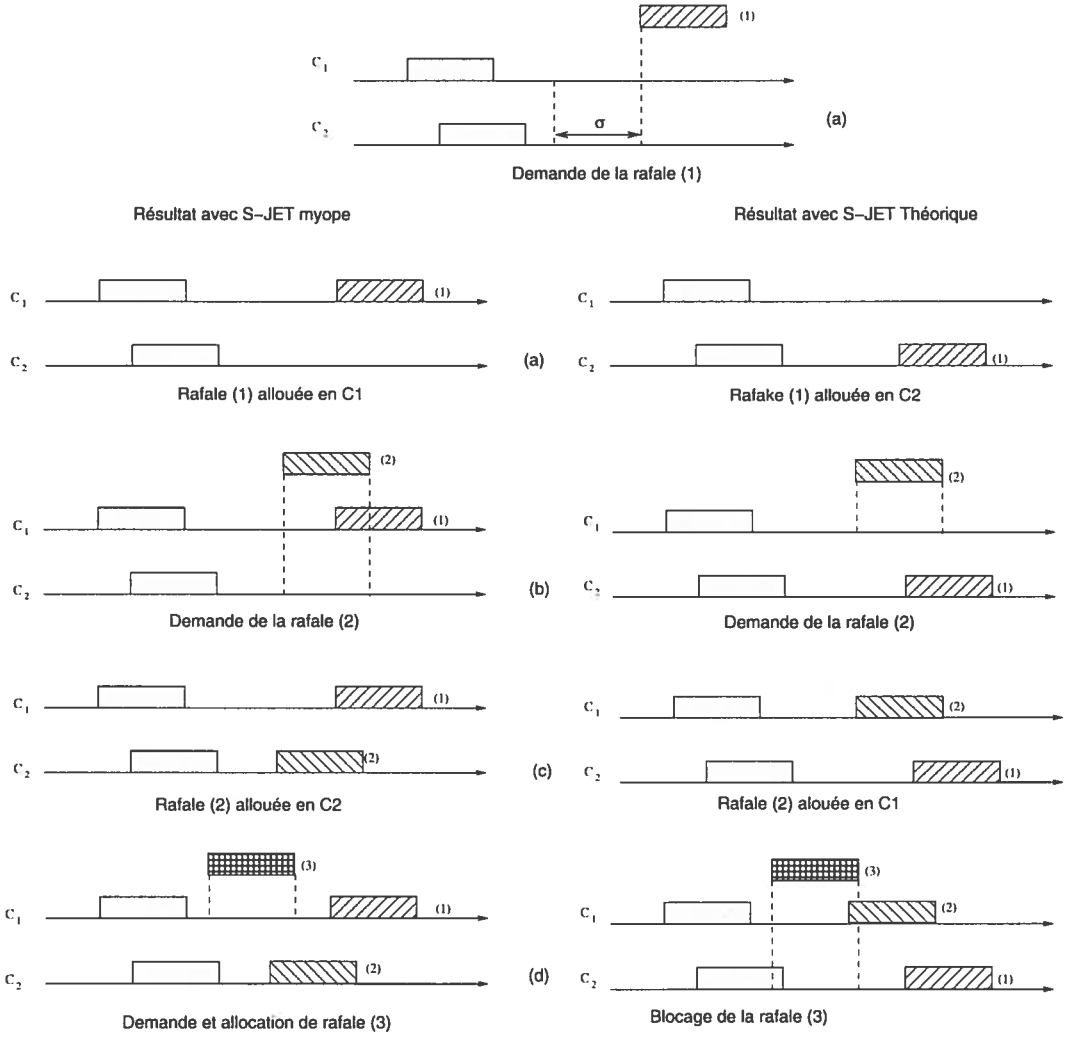



FIG. 3.6 – Blocage d'une réservation avec S-JET *théorique*

dépend de la manière que celles-ci sont assemblées par les couches plus élevées. Cependant, cette hypothèse ne compromet pas la validité des résultats car ils sont basés sur une comparaison d'autres algorithmes de réservation également évalués en simulant des arrivées de Poisson [5, 8, 12, 29].

**Proposition 1.** *Si le processus d'arrivée des rafales est Poisson* $(\lambda)$ *et que ses tailles sont identiquement et indépendamment distribuées avec une distribution générale de taille maximale* $\bar{b} < \infty$*, alors l'erreur de discrétisation due à l'algorithme de*

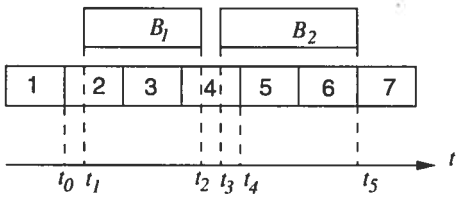



FIG. 3.7 – Une source additionnelle de blocage pour S-JET

*réservation ci-dessus mentionné diminue à zéro lorsque* $\Gamma \rightarrow 0$, *ayant pour résultat le processus d'attribution défini par les équations (2.3) – (2.5).*

*Démonstration.* La perte d'une rafale se produit lorsque l'algorithme S-JET ne trouve aucun canal libre. Toutefois, il existe une condition particulière où S-JET pourrait également bloquer une rafale mais cette fois-ci dû à la discrétisation de canaux. Puisque le processus d'arrivée $\{T_i\}$ est un processus Poisson homogène, en utilisant le principe PASTA de [18], le blocage dû à cet événement se produit lorsqu'il y a un seul canal candidat et que la demande d'une réservation commence ou se termine dans une case partiellement occupée.

Nous illustrons cet événement à l'aide de la figure 3.7. La réservation de la rafale $B_2$ commence à partir de l'instant $t_3$, ce qui correspond à la case 4. Cette case est partiellement occupée par la rafale $B_1$ jusqu'à l'instant $t_2$. Comme S-JET utilise la case entière, la rafale $B_2$ est bloquée. La probabilité de blocage due à la discrétisation est donc liée à la probabilité qu'une arrivée Poisson soit dans l'espace libre d'une case partiellement occupée, multipliée par la probabilité qu'un seul canal soit disponible pour allouer cette rafale.

Soit la variable aléatoire $\xi_i$, le point de départ des rafales consécutives dans l'intervalle $(0, t) \forall i = 1, \ldots, n$ sur un canal quelconque. Le processus d'arrivée des rafales est un processus Poisson de taux $\lambda$, et la taille des rafales est notée par $b_i$. Soit

$\{\xi_i\}$ une séquence d'événements dénotant le processus de départ d'un système de file d'attente de type $M/G/\infty$. Utilisant les méthodes que l'on retrouve au chapitre V section 4 de Taylor *et al* [21], nous obtenons que la distribution stationnaire du processus de départ est également un processus de Poisson. Soit $M(t)$ une variable qui compte le nombre de points de départ des rafales, c'est-à-dire :

$$M(t) = \sum_{n=1}^{N(t)} \mathbf{1}_{\{T_i + b_i \leq t\}},$$

de sorte que $\mathsf{P}(M(t) = m) = \mathsf{E}[\mathsf{P}(M(t) = m \,|\, N(t))]$. Du théorème 4.1 du chapitre mentionné précédement [21], la distribution conditionnelle de $T_i$, étant donné $N(t) = n$, est la distribution jointe des variables aléatoires uniformes iid pour (0,t) rangées dans l'ordre. Ainsi, pour $t \geq \bar{b}$,

$$p_t = \mathsf{P}(T_i + b_i \leq t \,|\, N(t) = n) = \frac{1}{t} \int_0^t \mathsf{P}(b_i \leq t - u)\, du$$

est indépendant de $i$. Ainsi le nombre de départs de rafales dans l'intervalle $(0, t)$, étant donnée $n$ arrivées Poisson, est une variable aléatoire binomiale ayant comme paramètres $(n, p_t)$, de sorte que :

$$\mathsf{P}(M(t) = m) = \mathsf{E}[\mathrm{Bin}(N(t), p_t)] = \frac{e^{\lambda p_t t}(\lambda p_t t)^m}{m!},$$

prouvant que $M(t)$ est une variable aléatoire Poisson ayant une moyenne de $\lambda p_t t$. La preuve est completée en notant que

$$\lim_{t\to\infty} p_t = \lim_{t\to\infty} \frac{1}{t} \int_0^t G(t-u)\, du = \lim_{t\to\infty} \frac{1}{t} \int_0^t G(u)\, du = 1,$$

où $G(u)$ est la fonction de distribution de la taille de la rafale et elle satisfait $G(u) = 1, u \geq \bar{b}$. Ce résultat reste valide, même si la taille de la rafale n'est pas uniformément distribuée, tant qu'elle possède une variance finie.

En utilisant le même théorème 4.1 de Taylor *et al.* [21], nous concluons que la distribution de la taille de l'espace libre dans une case de taille $\Gamma$, est $\zeta_i = \Gamma \lceil \xi_i/\Gamma \rceil - \xi_i$ de distribution uniforme entre $(0, \Gamma)$, parce que $\{\xi_i\}$ est un processus de Poisson stationnaire. En effet, étant donné qu'il y a au moins un tel point dans une case donnée, la distribution conditionnelle du point réel où il se trouve est uniforme dans l'intervalle.

Selon la définition d'un processus de Poisson, la probabilité qu'une arrivée $A$ se trouve dans un intervalle $(\xi_i, \Gamma \lceil \xi_i/\Gamma \rceil]$ de petite taille $\zeta_i \approx 0$ est $\lambda\zeta_i + \mathcal{O}(\Gamma^2)$, de sorte que

$$\mathsf{P}(A \in (\xi_i, \Gamma \lceil \xi_i/\Gamma \rceil) \approx \mathsf{E}(\lambda\zeta_i) = \frac{\lambda\Gamma}{2},$$

ce qui démontre la proposition, parce que la probabilité de blocage due à la discrétisation est la probabilité que l'arrivée fasse partie de cet intervalle et qu'il n'y ait aucun autre canal disponible. La limite $\lambda\Gamma/2$ est plus serrée pour un nombre plus petit de canaux, et elle surestimera considérablement le blocage additionnel réel lorsque le nombre de canaux augmente. □

# Chapitre 4

# Analyse des simulations

Dans ce chapitre, nous présentons les résultats des simulations des algorithmes S-JET, JET et Horizon. Le critère de comparaison utilisé pour déterminer la performance des algorithmes est la probabilité de perte des rafales, définie par le rapport du nombre de rafales perdues sur le nombre de rafales total envoyées au commutateur. Dans la section 4.1, nous discutons des modèles utilisés pour évaluer la performance des algorithmes. Nous présentons deux scénarios : le premier utilisant des rafales ayant la même taille (section 4.1.1) et le deuxième utilisant des rafales de taille variable (section 4.1.2). La section 4.3 porte sur une étude permettant de déterminer le temps de traitement requis par chacun des algorithmes. Les résultats de nos simulations sont énoncés à la section 4.4. Nous présentons la performance de S-JET lorsque la taille des rafales est constante, section 4.4.1, et lorsque la taille des rafales est variable, section 4.4.2. Finalement nous commentons dans la section 4.5, nos résultats et nous discutons des conditions idéales favorisant la performance de S-JET.

## 4.1 Modèle de simulation

Pour évaluer la performance des algorithmes de réservation, nous nous concentrons sur un seul commutateur. Ce commutateur possède $C$ canaux de sortie pour l'envoi des données et un canal exclusif pour les en-têtes de contrôle. Il fournit la pleine conversion de longueurs d'onde. Nous supposons que les en-têtes de contrôle ne sont jamais perdus. Pour une rafale quelconque, le nombre de sauts, notée $H$, ainsi que le temps de traitement pris par chaque algorithme au commutateur ($\tau$), sont utilisés pour déterminer la valeur finale du décalage de la rafale. La variable $H$ est une variable aléatoire uniforme discrète sur l'intervalle [1,10]. En termes mathématiques, le décalage $\delta$ est égal à : $H\tau$.

### 4.1.1 1 : Rafales à taille constante

Nous allons tout d'abord étudier un commutateur vers lequel des rafales de même taille arrivent. Nous voulons évaluer la probabilité de perte des rafales dans le commutateur en question. Si $x$ représente le nombre de rafales perdues et $y$ représente le nombre de rafales générées, alors la probabilité de perte est donnée par $x/y$. Le processus d'arrivée des rafales est un processus de Poisson de taux $\lambda$. Dans ce modèle la taille des rafales est fixe.

Pour estimer la probabilité de perte d'une rafale dans un commutateur, le processus d'arrivée des rafales est fréquemment modélisé par un processus Poisson [5, 8, 12, 29].

### 4.1.2 2 : Rafales de taille variable

Le deuxième modèle considère des rafales de petite, moyenne et grande taille. Pour chaque taille, nous voulons évaluer la perte des rafales dans le commutateur en question. Les processus d'arrivée sont modélisés par des processus Poisson de taux $\lambda_p$, $\lambda_m$ et $\lambda_g$, où $\lambda_p$ représente le taux d'arrivé des petites rafales, $\lambda_m$ celui des moyennes rafales et $\lambda_g$ celui des grandes rafales. Nous considérons que tous les processus d'arrivée des rafales au commutateur sont indépendants. Nous supposons que toutes les rafales appartiennent à la même classe de service.

Dans nos programmes de simulation, nous générons un processus Poisson de taux $\bar{\lambda} = \lambda_p + \lambda_m + \lambda_g$. Lorsque la rafale arrive au commutateur, sa taille est établie de la façon suivante :

- $P$(taille de la rafale est petite) = $\lambda_p/\bar{\lambda}$
- $P$(taille de la rafale est moyenne) =$\lambda_m/\bar{\lambda}$
- $P$(taille de la rafale est grande) = $\lambda_g/\bar{\lambda}$

Nous travaillons avec trois scénarios différents pour évaluer la performance des algorithmes. Nous varions le taux d'arrivée des rafales de chaque taille comme suit :

- Scénario 1 : $\lambda_p = 0.40\bar{\lambda}, \lambda_m = 0.10\bar{\lambda}, \lambda_g = 0.50\bar{\lambda}$
- Scénario 2 : $\lambda_p = 0.45\bar{\lambda}, \lambda_m = 0.10\bar{\lambda}, \lambda_g = 0.40\bar{\lambda}$
- Scénario 3 : $\lambda_p = 0.40\bar{\lambda}, \lambda_m = 0.40\bar{\lambda}, \lambda_g = 0.20\bar{\lambda}$

## 4.2 Calcul de $\Gamma$ et $M$

Le paramètre de base pour notre algorithme est la taille d'une case, notée $\Gamma$. Nous suggérons d'exprimer ce paramètre en fonction des quantités de base dans

le système, c'est-à-dire $\Gamma = \bar{b}/M$ où $M$ est le nombre de cases nécessaires pour représenter une rafale ayant la longueur maximale $\bar{b}$.

Nous avons réalisé de nombreuses simulations pour différents systèmes afin de choisir la valeur pour le paramètre $M$ ayant la meilleure performance. Le modèle utilisé dans nos simulations est semblable à celui de Vázquez-Abad *et al.* [23] dans lequel le décalage est determiné au commutateur d'entrée et fixé à $H\tau$. Nous allons considérer que le temps de traitement est le même pour tous les algorithmes. Ces expériences montrent que, parmi les valeurs testés pour le paramètre $M$, la valeur qui donne une probabilité de blocage pour S-JET la plus rapprochée de celle de JET est $M = 64$. Dans le but de justifier cette affirmation, la figure 4.1 illustre la probabilité de blocage obtenue pour différentes valeurs de $M$ en utilisant les paramètres du tableau 4.1.

| *Paramètre* | *Valeur* |
|---|---|
| Canaux | 4 |
| $\bar{b}$ | $3\mu$s |
| Charge du lien | 0.25 |
| $\tau$ | $1\mu$s |
| $M$ | 20 – 64 |

TAB. 4.1 – Paramètres de la simulation

Nous observons que le nombre de cases par rafale joue un rôle important dans la probabilité de perte globale. Pour la méthode S-JET, nous constatons qu'une diminution de la taille d'une case ($\Gamma$) entraîne une diminution de la probabilité de blocage.

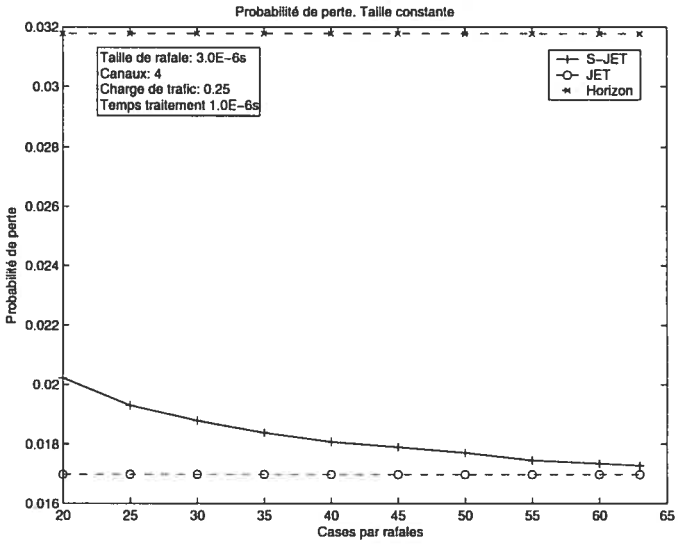



FIG. 4.1 – Performance de S-JET en utilisant différentes valeurs pour $M$

## 4.3 Complexité des algorithmes

Selon Vázquez-Abad *et al.* [23], le temps de traitement requis par les algorithmes affecte considérablement la probabilité de perte. Pour analyser les trois algorithmes, il est essentiel de tenir compte de ce temps de traitement pour assigner une rafale. Afin d'estimer le temps de traitement d'un algorithme, nous avons programmé une boucle exécutant celui-ci un million de fois. Nous utilisons les *tics* de CPU comme paramètre de base pour mesurer le temps de traitement. Un tic est la plus petite unité de temps reconnu par le CPU. Un tic, typiquement, prend entre 1/100 à 1/10ème de seconde [30]. Le tic est simplement limité par la puissance du CPU, plus le processeur est rapide, moins le tic durera. Le nombre de tics pris par chacun des algorithmes se retrouvent au tableau 4.2. L'ordinateur utilisé pour les tests est un AuthenticAMD avec un processeur AMD Athlon(tm) XP 2800+ à 2106.499 MHz.

Tel que prévu (sections 2.1.1 et 2.1.2), les tics CPU de JET changent selon le trafic, tandis que Horizon et S-JET demeurent stables, puisque le calcul effectué

| *Algorithme* | *Tics CPU* |
|---|---|
| S-JET | 410 |
| Horizon | 33 |
| JET | 13621 à 23737 |

TAB. 4.2 – Temps de traitement en tics de CPU requis par les algorithmes

par Horizon dépend uniquement du nombre de canaux et qu'il n'est pas affecté par le trafic. Le calcul effectué par JET est affecté, non seulement par le nombre de canaux, mais aussi par la charge (trafic) du réseau. Plus la charge par lien augmente, plus le nombre de recherches par canal augmente. Rappelons que JET effectue une recherche en deux dimensions sur une liste (équation 2.4). Pour cette raison le nombre de tics varie en fonction de la charge. Cette variation est démontrée empiriquement par les résultats de nos simulations (Tableau 4.2). En ce qui concerne S-JET, le calcul effectué pour trouver un espace libre est lié au nombre de canaux. De façon similaire à Horizon, une seule comparaison par canal est requise pour trouver les canaux candidats (fig. 3.2), à cause de la discrétisation des rafales et du temps de réservation. La charge par lien n'est donc pas un facteur déterminant affectant le temps de traitement.

D'après les informations contenues au tableau 4.2, nous obtenons l'approximation suivante pour le temps de traitement : soit $\theta$ le nombre de tics d'un algorithme donné et $f$ la fréquence de l'unité centrale en mégahertz (Mhz). Le temps de traitement $\tau$ au contrôleur en microsecondes (ms) s'exprime comme suit :

$$\tau = \Delta + \frac{\theta \times 10^6}{f}, \tag{4.1}$$

où $\Delta$ est le temps exigé par chaque commutateur pour traiter la longueur d'un entête. Dans nos programmes de simulation, nous supposons que $\Delta = 1\mu s$ et nous

utilisons le modèle de [23] qui propose de fixer le décalage initial à $H\tau$ dans le but d'empêcher des arrivées précoces. Ceci affecte la distribution du décalage à chaque commutateur (figure 2.1). Les valeurs du paramètre $\tau$ sont affichées au tableau 4.3.

| *Algorithme* | $\tau$ ($\mu$s) |
|---|---|
| Horizon | 1.01563 |
| S-JET | 1.19463 |
| JET | 9.7158 - 12.268 |

TAB. 4.3 – Temps de traitement $\tau$ (en microsecondes) requis par les algorithmes

Même si le temps de traitement de S-JET est 13 fois plus élevé que celui d'Horizon, cet algorithme s'avère un choix intéressant puisqu'il offre une faible probabilité de perte et qu'il permet d'implanter des mécanismes de qualité de service.

## 4.4 Comparaison de la probabilité de perte

Nous comparons la performance des algorithmes avec différentes conditions de la charge du trafic. Pour les deux scénarios étudiés, la charge du trafic, $\rho$, s'obtient par la relation suivante [12] :

$$\rho = \frac{\lambda \bar{b}}{C} \tag{4.2}$$

Les valeurs utilisées pour la charge du trafic sont comprises entre 0.2 et 0.5. Nous utilisons les valeurs suivantes pour le nombre de canaux au commutateur et le nombre de sauts : $C = 4$, $H \in \{1, \ldots, 10\}$.

Nous évaluons la performance de S-JET en utilisant différentes valeurs pour les paramètres $\bar{b}$, $\rho$ et $\tau$ (tableau 4.3). Un nombre suffisamment grand de réplications dans la simulation permet de donner une estimation par intervalle de confiance de

95% à une précision raisonnable. Nous montrons à l'aide des résultats de simulation que S-JET possède un rendement satisfaisant par rapport à celui de Horizon.

### 4.4.1 Probabilité de perte avec rafales de taille constante

Tel que démontré par Vázquez-Abad *et al.* [23], en raison du temps de traitement pris par JET, Horizon peut donner une meilleure performance que JET. Ce délai supplémentaire de JET augmente la variance du décalage des en-têtes qui arrivent au commutateur. Rappelons qu'après avoir subi la conversion optoélectronique et après l'exécution de l'algorithme de réservation, le décalage ($\delta$) de l'en-tête de contrôle sera réduit de $\tau$ unités de temps. Ainsi, les valeurs de $\tau$ pour JET ne sont pas constantes comme dans le cas d'Horizon (tableau 4.3).

La situation réelle est encore moins avantageuse que les résultats obtenus par simulation. Si plusieurs demandes sont traitées simultanément, alors le temps de traitement élevé de JET peut engendrer la création d'une file d'attente au contrôleur et l'apparition d'arrivées précoces : ces deux conséquences indésirables peuvent être la source de blocages supplémentaires. S-JET améliore l'utilisation des canaux par rapport à Horizon, car il a un temps de traitement plus court que JET, rendant ainsi la variance des décalages semblable à celle de l'algorithme Horizon.

La figure 4.2 montre les courbes obtenues en utilisant des rafales de taille constante $\bar{b} = 10\mu$s. Cette expérience illustre clairement que la performance de l'algorithme S-JET surpasse la performance de l'algorithme JET. Nous verrons à la prochaine section que l'utilisation de S-JET peut présenter des avantages additionnels lorsque la taille de la rafale n'est pas constante.

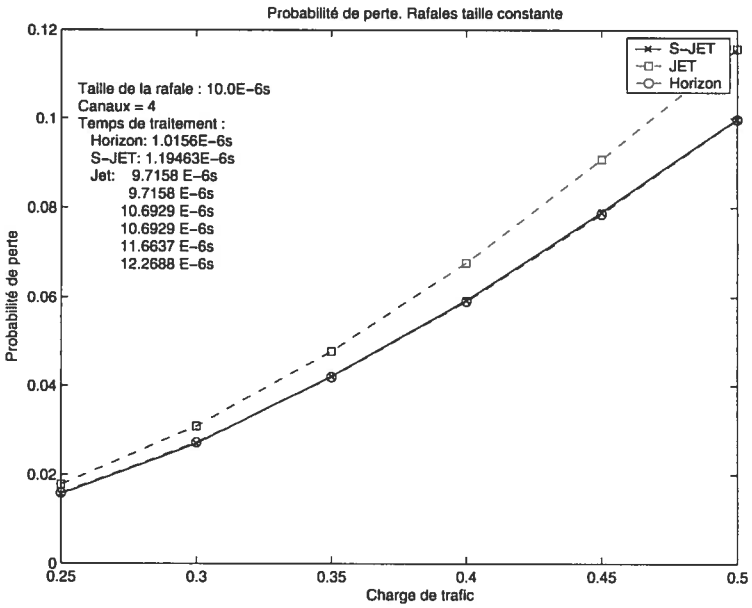



FIG. 4.2 – Probabilité de blocage pour Horizon, JET et S-JET

### 4.4.2 Probabilité de perte avec rafales de taille variable

Nous utilisons maintenant des rafales de taille variable, petite, moyenne ou grande. Nous désirons évaluer la probabilité de perte en fonction de la taille des rafales. Pour l'algorithme JET, plus la taille d'une rafale est petite, plus les chances sont élevées de trouver un espace vide pour cette rafale. Ainsi, nous prévoyons que la probabilité de perte de rafales de petite taille sera considérablement plus faible que celle des rafales de tailles moyenne ou grande. Les algorithmes remplissant les vides, comme JET et S-JET, peuvent accorder des priorités aux rafales dont la taille est petite. Ceci nous aidera à déterminer si la taille d'une rafale peut être utilisée comme mécanisme de QS.

Pour les trois scénarios choisis précédemment, les tailles des petites, moyennes et grandes rafales sont indiquées dans le tableau 4.4. Nous considérons que toutes les rafales appartiennent à la même classe. La charge par lien est calculée selon

l'équation (4.2). Étant donné les résultats obtenus à la section précédente avec JET qui montrent que cet algorithme est dominé par S-JET, nous nous limitons ici à présenter des résultats avec Horizon et S-JET.

| *Type de rafale* | *Taille* |
|---|---|
| Petite | $2\mu$s |
| Moyenne | $3\mu$s |
| Grande | $10\mu$s |

TAB. 4.4 – Taille de la rafale

### Scénario 1

Pour le commutateur en question, soit $l_p$ le nombre de petites rafales perdues, $l_m$ celui de moyennes rafales perdues, et $l_g$ celui de grandes rafales perdues. Si $y$ représente le nombre de rafales observées demandant une réservation, alors, la probabilité de perte globale s'obtient par la relation suivante : $(l_p + l_m + l_g)/y$.

La figure 4.3 montre la probabilité de perte globale obtenue par les deux algorithmes, pour le scénario 1. Les résultats illustrent clairement que la probabilité de perte de S-JET est inférieure à celle de Horizon. Cette différence est plus évidente à mesure que la charge par lien augmente. En d'autres termes, la majorité des longueurs d'onde ont un taux d'utilisation élevé en raison de la charge de trafic. Par conséquent, il est moins probable qu'une rafale puisse trouver une longueur d'onde disponible.

Dans la figure 4.4, nous montrons la probabilité de perte avec S-JET pour les trois tailles considérées. Nous constatons qu'il est possible de différencier les rafales qui sont perdues de celles qui ne le sont pas. Avec S-JET, la probabilité de perte des

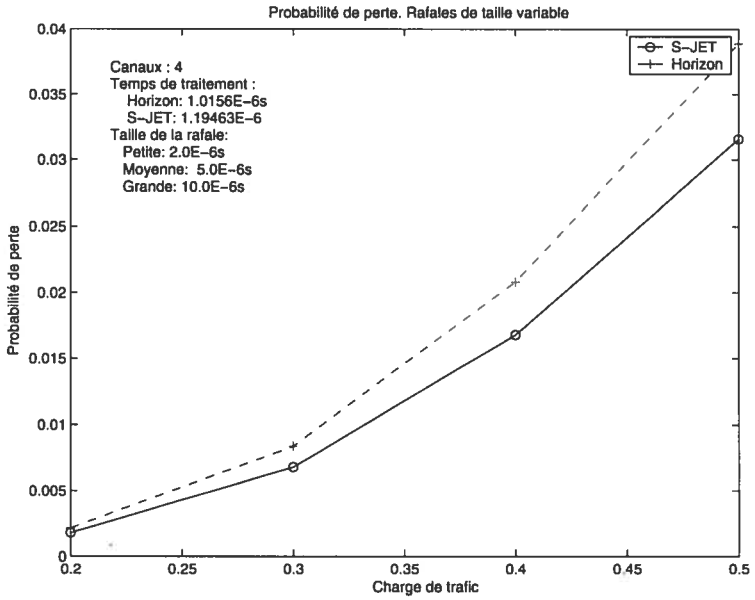



FIG. 4.3 – Probabilité de perte globale de Horizon et S-JET, scénario 1

petites rafales est faible par rapport à la probabilité de perte des grandes rafales. Plus la charge par lien augmente, plus l'écart entre les probabilités de perte des petites et des grandes rafales est élevé. Il est intéressant de noter que la perte obtenue avec l'algorithme Horizon est comparable au pire cas de S-JET, c'est-à-dire celui obtenu avec les grandes rafales.

Nous ne montrons pas les résultats de Horizon, puisque le rapport du nombre de rafales envoyées contre les rafales bloquées demeure le même pour chaque catégorie. La probabilité de perte de Horizon est indépendante de la taille des rafales puisque cet algorithme effectue une recherche seulement à la fin de la liste. La performance d'Horizon n'est pas affectée lorsque la taille des rafales varie.

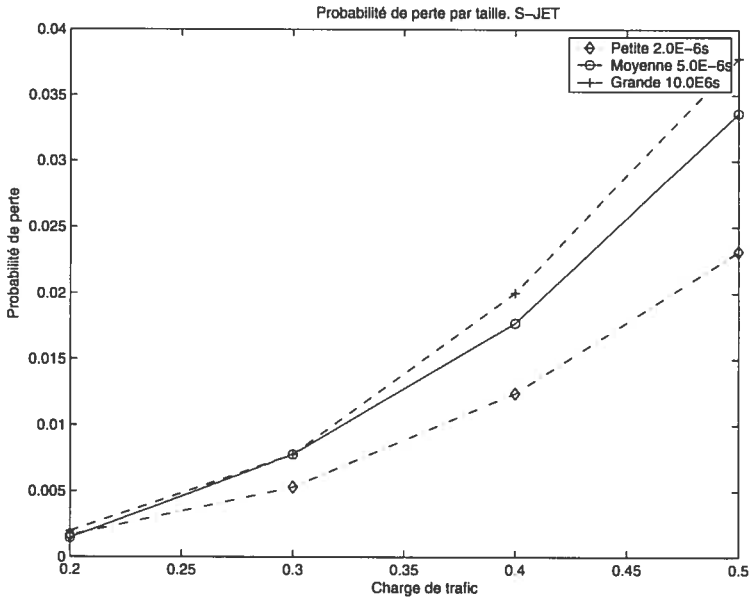



FIG. 4.4 – Probabilité de perte par taille S-JET

### Scénario 2

Les figures 4.5 et 4.6 présentent les résultats du deuxième scénario. Les proportions espérées des petites, moyennes et des grandes rafales sont respectivement 45%, 10% et 45%. Nous pouvons voir que le comportement des courbes est semblable à celui observé avec le scénario 1.

### Scénario 3

Nous montrons aux figures 4.7 et 4.8 les résultats de la simulation où le pourcentage de grandes rafales comporte 20% du trafic tandis que le pourcentage de petites et moyennes rafales est de 40%. La performance de S-JET est grandement ameliorée, comparée à celle de Horizon, lorsque nous réduisons le pourcentage des grandes rafales.

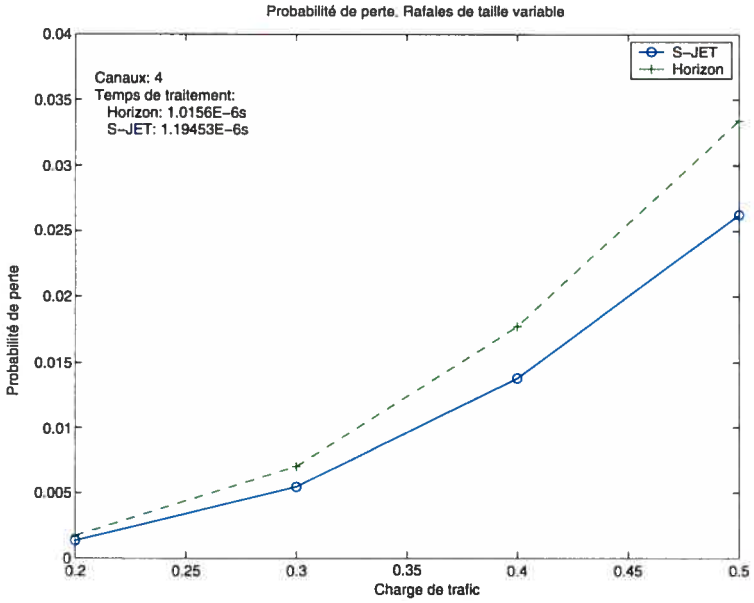



FIG. 4.5 – Probabilité de perte globale Horizon et S-JET, scénario 2

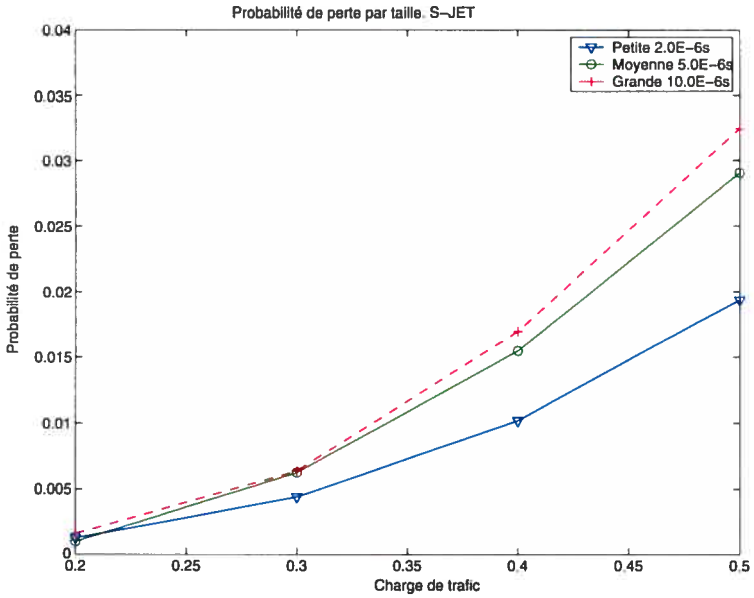



FIG. 4.6 – Probabilité de perte par taille S-JET, scénario 2

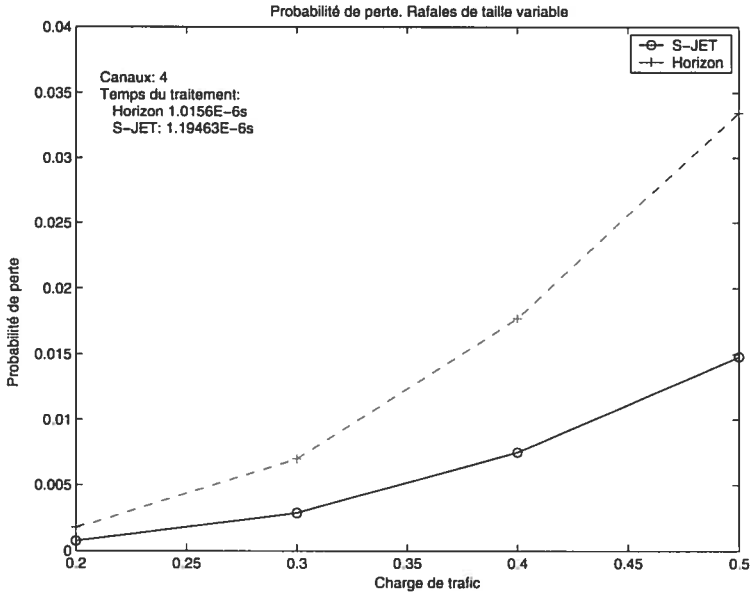



FIG. 4.7 – Probabilité de perte globale de Horizon et S-JET, scénario 3

Nous observons que S-JET a un meilleur rendement lorsqu'il y a des intervalles libres à remplir. Pour favoriser la création de ces intervalles, une valeur suffisamment grande pour le paramètre $H$ ou le paramètre $\delta$ par rapport à la taille de la rafale est nécessaire. Cependant, une valeur élevée du paramètre $H$ impliquerait un nombre élevé de commutateurs à traverser par la rafale. Cette situation n'est pas souhaitable puisque nous préférons avoir le nombre de commutateurs le plus réduit à l'intérieur du réseau. L'autre façon de favoriser la création des intervalles libres est donc l'utilisation du décalage. Nous avons également observé que S-JET offre une faible probabilité de perte lorsqu'il est employé dans un environnement où les rafales ont plusieurs tailles.

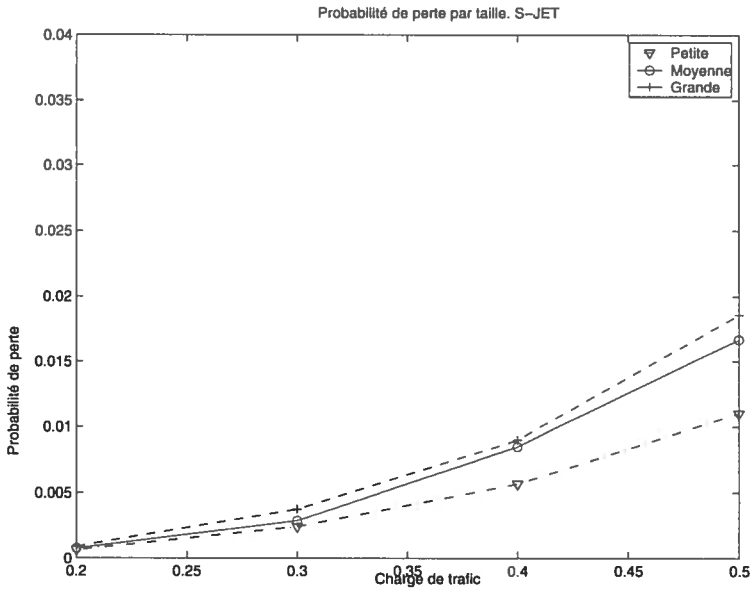



FIG. 4.8 – Probabilité de perte par taille S-JET, scénario 3

## 4.5 Observations

L'architecture de réseau OBS a été proposée comme un modèle pour transporter le trafic de paquets IP directement dans un réseau optique à commutation d'onde dense [4, 10, 24]. Cependant, le protocole IP a été conçu sans tenir compte que certains paquets exigeaient un traitement spécial. Puisque ce protocole ne fait pas la distinction entre les différents types de paquets, même si pour certains paquets le temps de livraison est crucial, il offre un service appelé *best effort* [11]. Ainsi, l'intégration de mécanismes de qualité de service (QS) est importante afin d'assurer que les rafales, et par conséquent les paquets, soient livrés selon certains critères. Pour ce faire, des mécanismes de qualité de service ont été développés. Ces mécanismes permettent de faire la distinction entre les différents types de paquets, traitant ainsi les paquets selon les paramètres définis par les opérateurs du réseau. Leur performance est évaluée par rapport à certains facteurs tels que le temps de propagation, la gigue,

la probabilité de perte, etc. Un exemple d'un tel mécanisme est *DiffServ* [2]. Celui-ci fonctionne en se servant d'un des champs se trouvant dans l'en-tête de contrôle de chaque paquet IP. En lisant l'information stockée dans ce champ, les commutateurs sont capables d'identifier le type du paquet reçu et de le traiter selon les critères spécifiques associés à ce type. Le protocole MPLS [17] représente un autre mécanisme utilisé pour fournir la qualité de service. Dans ce cas-ci, une étiquette est ajoutée à chaque paquet IP afin de le différencier et de lui assigner un chemin de livraison. Les paquets sont envoyés à travers un ou plusieurs chemins étiquetés (LSP) établis à l'avance par les commutateurs. L'utilisation des étiquettes permet de prendre des décisions de routage en se basant uniquement sur leur valeur et de ne pas effectuer de calculs complexes de routage portant sur l'adresse IP au niveau de la couche 3 du modèle TCP/IP.

Bien que MPLS et DiffServ soient couramment utilisés pour les réseaux IP, leur implantation comme mécanismes de qualité de service pour les réseaux OBS engendre beaucoup de difficultés. D'un côté, OBS emploie des rafales contenants plusieurs paquets. Puisque les rafales ne sont pas lues aux commutateurs intermédiaires, ceux-ci ne peuvent pas identifier les paquets contenus dans les rafales. D'un autre côté, puisque les paquets peuvent avoir un délai additionnel, l'architecture des commutateurs OBS ne considère pas l'utilisation de mémoire tampon (fig. 1.6), ce qui rend impossible l'emploi de fibres de délai. Les mémoires tampon sont indispensables pour implanter certains mécanismes de QS, dont Diffserv.

Par conséquent, la conception de nouveaux mécanismes permettant d'offrir une qualité de service dans les réseaux OBS joue un rôle primordial dans leur conception. Dans ce sens, nous considérons que l'algorithme S-JET peut être utilisé non

seulement pour la gestion de réservation aux commutateurs optiques mais également comme un système de support aux mécanismes de qualité de service pour les réseaux OBS. Dans la littérature plusieurs mécanismes ont été proposés dont deux ont attiré notre attention : le mécanismes qui utilisent le décalage et ceux qui utilisent la segmentation.

Chen *et al.* [3] présentent une méthode basée sur le décalage. Cette technique consiste à utiliser le décalage afin d'isoler les rafales de classes différentes. Les rafales qui appartiennent à une classe prioritaire auront un décalage plus grand et par conséquent, leur probabilité de perte sera réduite. L'autre méthode que nous allons mentionner est présentée dans la littérature comme le modèle de segmentation [24]. Avec la méthode de segmentation, lorsqu'une collision se produit, la partie de la rafale qui chevauchait la rafale déjà allouée est segmentée, pour ensuite être jetée ou re-acheminée. L'autre partie est donc allouée sur le canal.

Ces méthodes offrent des conditions idéales pour l'implantation de l'algorithme S-JET : la création des intervalles libres à cause du décalage additionnel utilisé par le modèle de [3] ainsi que la création de rafales de plusieurs tailles à cause de la segmentation [24].

# Conclusion

Dans ce mémoire nous avons présenté S-JET, un algorithme pour la gestion de réservation pour l'architecture de réseaux OBS. Notre algorithme utilise le principe «void filling»pour optimiser l'utilisation de la bande passante dans un canal, tout en ayant une faible complexité informatique. S-JET ne requiert pas un temps de traitement élevé dans le contrôleur, lui permettant d'offrir une faible probabilité de perte. Nous avons effectué une étude expérimentale en utilisant différents scénarios. Pour chaque scénario, nous avons fait varier la taille de la rafale, le nombre de canaux ainsi que le temps de traitement au contrôleur. Nous avons comparé le temps de traitement et la probabilité de perte entre Horizon, JET et S-JET. Nos expériences ont démontré que S-JET a une complexité informatique moins lourde que celle de JET tout en offrant une probabilité de perte plus avantageuse que celle de Horizon sous certaines conditions. En utilisant des opérations booléennes, il détermine s'il y a un intervalle libre disponible pour allouer la rafale dans un des canaux au port de sortie. Dans le pire des, cas nous avons observé une probabilité de perte similaire entre l'algorithme Horizon et l'algorithme S-JET.

Nous avons aussi proposé l'utilisation de S-JET dans le cas d'une différentiation selon la taille de la rafale. Les résultats de nos expériences indiquent que l'implantation de l'algorithme S-JET est une alternative intéressante pour les réseaux OBS,

spécialement si S-JET s'utilise dans un environnement favorisant la création d'intervalles vides ou dans un environnement ayant des rafales de plusieurs tailles.

À partir de ces résultats nous avons obtenu un brevet provisoire pour S-JET en novembre 2004 [19]. Nous considérons qu'il est possible d'aller plus loin dans le développement de S-JET, au-delà d'un mécanisme de réservation. Il est important de souligner qu'il existe d'autres facteurs intéressants à explorer comme l'implantation de mécanismes de qualité de service dans les réseaux OBS, l'assemblage de rafales aux noeuds d'accès du réseau et la façon de gérer les collisions entre les rafales pour réduire la probabilité de perte. Nous croyons que notre algorithme, combiné avec un ou plusieurs de ces mécanismes, peut offrir un modèle de réseau attrayant tant pour les fournisseurs de services de réseau que pour les utilisateurs.

# Bibliographie


[1] T. Battestilli, H. Perros, *"An introduction to Optical Burst Switching,"* IEEE Optical Communications, pp. 510–515, Août 2003.

[2] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, *"An architecture for Differentiated Services,"* RFC 2475, Dec. 1998.

[3] Y. Chen, M. Hamdi, D.H.K. Tsang, C. Qiao, *"Providing Proportionally Differentiated Services over Optical Burst Switching Networks,"* Proceedings of IEEE Globecom, vol. 3, pp. 1510-1514, 2001.

[4] Y. Chen, C. Qiao, X. Yu, *"Optical Burst Switching : A New Area in Optical Networking Research,"* IEEE Network, vol. 18, no. 3, pp. 16-23, Mai/Juin 2004.

[5] Y. Chen, H. Wu, D. Xu, C. Qiao, *"Performance analysis of Optical Burst Switched Node with deflection routing,"* ICC 2003 - IEEE International Conference on Communications, vol. 26, no. 1, pp. 1355-1359, Mai 2003.

[6] H.M Deitel, P.J. Deitel, *Comment programmer en Java™* 3e. Édition, Les éditions Reynald Goulet Inc., 2000.

[7] K. Dolzer, C. Gauger, J. Spath, S. Bodamer, *"Evaluation of Reservation Mechanisms for Optical Burst Switching."* AEU International Journal of Electronic Communications, vol. 55, no. 1, 2001.

[8] A.M. Law, W.D. Kelton, *Simulation Modeling and Analysis*, McGraw-Hill Higher Education, 2000.

[9] P. L'Ecuyer, L. Meliani, J. Vaucher, *"SSJ : A framework for stochastic simulation in Java"*, Proceedings 2002 Winter Simulation Conference, IEEE press, pp. 234–242. 2002.

[10] M. Listanti, V. Eramo, R. Sabella, *"Architectural and Technological Issues for Future Optical Internet Networks."* IEEE Communication Magazine, pp. 82-92, Sep. 2000.

[11] C. Loi, Q. Liao, D. Yang. *"Service Differentiation in Optical Burst Switched Networks"*, Proceedings IEEE Globecom '02, vol. 21, no. 1, pp. 2325-2329, Nov. 2002.

[12] M. Neuts, Z. Rosberg, H. L. Vu, J. White, M. Zukerman, *"Performance enhancement of optical burst switching using burst segmentation,"* Proceedings of IEEE ICC, vol. 3, pp. 1828-1832, 2003.

[13] C. Qiao, *"Labeled optical burst switching for IP-over-WDM integration,"* IEEE Communications Magazine, vol. 38, pp. 104–114, Sep. 2000.

[14] C. Qiao, M. Yoo, *"Optical burst switching (OBS) – a new paradigm for an optical internet,"* J. High Speed Networks, vol. 8, pp. 69–84, Jan. 1999.

[15] J. Ramamirtham, J. Turner, *"Time-Sliced Optical Burst Switching,"* Proceedings of IEEE Infocom 2003, vol. 22, no. 1, pp. 2030-2038, Mars 2003.

[16] R. Ramaswami, K.N. Sivarajan, *Optical Networks. A practical perspective*, Morgan Kaufmann Publishers, Inc., San Francisco, 1998.

[17] E. Rosen, A. Viswanathan, R. Callon, *"Multiprotocol Label Switching Architecture,"* RFC 3031, Jan. 2001.

[18] S.M. Ross, *Introduction to probability models*, 8e. Édition, Academic Press, Inc. 2002

[19] B. Sansò, F.J. Vázquez-Abad, E. Gutiérrez-Cabrera, *S-JET : An efficient reservation scheduling algorithm for Optical Burst Switches.* "Provisional US patent filed on a new scheduling method for burst switching." Nov. 15, 2004. No. 60/627,286.

[20] A. Tanenbaum, *Réseaux*, 3e. Édition, Prentice Hall, 1996.

[21] H.M. Taylor, S. Karlin, *An Introduction to Stochastic Modeling*, 3e. Édition, Academic Press, San Diego, 1998.

[22] J. Turner, *"Terabit burst switching,"* Journal of High Speed Networks, vol. 8, pp. 3–16, Jan. 1999.

[23] F.J. Vázquez-Abad, J. A. White, L.L.H. Andrew, R. S. Tucker, *"Does header length affect performance in optical burst switched networks ?,"* Journal of Optical Networking, Vol. 3, No. 5, pp. 342–362, Mai 2004.

[24] J.M. Vokkarane, J.P. Jue, *"Prioritized Burst Segmentation and Composite Burst-Assembly Techniques for QoS Support in Optical Burst-Switched Networks",* IEEE Journal on Selected Areas in Communication, Vol. 21 No. 7, pp. 1198-1209, Sep. 2003.

[25] J. Xu, C. Qiao, J. Li, G. Xu, *"Efficient Channel Scheduling Algorithms in Optical Burst Switched Networks",* IEEE Infocom 2003, vol. 22, no. 1, pp. 2268-2278, Mars 2003.

[26] L. Xu, H. Perros, G. Rouskas, *"Techniques for Optical Packet Switching and Optical Burst Switching,"* IEEE Communication Magazine, pp. 136–142, Jan. 2001.

[27] Y. Xiong, M. Vandenhoute, H. Cankaya, *"Control Architecture in Optical Burst-Switched WDM Networks"* IEEE Journal on Selected Areas in Communications, pp. 1838-1851, Oct. 2000

[28] M. Yoo, C. Qiao, S. Dixit, *"QoS Performance of Optical Burst Switching in IP-Over-WDM Networks,"* IEEE Journal on Selected Areas in Communication Vol. 18, No. 10, pp. 2062-2071, Oct. 2000.

[29] X. Yu, Y. Chen, C. Qiao, *"Study of traffic statistics of assembled burst traffic in optical burst switched networks."* Proceedings of Opticomm, pp. 149-159, 2002.

[30] http ://www.dicofr.com, *"Dictionnaire de l'informatique et d'internet"*

# Annexe A

# T-SIM : simulateur de réseaux optiques

## A.1 Introduction

Le but de ce chapitre est d'esquisser la modélisation des modules composant le logiciel de simulation T-SIM, ainsi que de présenter le schéma général du fonctionnement du simulateur. T-SIM a été développé dans un langage de programmation généraliste (Java) en s'appuyant sur l'environnement de simulation stochastique en Java (SSJ) [9] pour la gestion des événements et la génération des nombres aléatoires. Ce logiciel a été développé au laboratoire d'optimisation et de simulation de l'Université de Montréal en collaboration avec Jolyon Whyte de l'Université de Melbourne en Australie. Dans sa première étape, T-SIM vise à offrir un outil de simulation spécialement conçu pour les réseaux OBS. Dans les étapes ultérieures, nous allons introduire des classes Java additionnelles pour répondre aux besoins de simulation d'autres réseaux de télécommunication.

## A.2 Fonctionnement de T-SIM

La première version de T-SIM est formée de deux modules : le premier module, T-SIM, représente le noyau du simulateur. Le deuxième module, OBS, est utilisé pour simuler les réseaux OBS. Notre équipe de recherche est aussi en train de développer des modules additionnels pour simuler d'autres types de modèles de réseaux (IP, ATM, etc.). Dans la figure A.1 nous illustrons la structure de T-SIM.


FIG. A.1 – Structure de T-SIM

### A.2.1 Le noyau de T-SIM

Les classes et méthodes dans ce module fournissent les éléments nécessaires pour capturer le comportement des objets dans le modèle de simulation à utiliser (voir figures A.3, A.4 et A.5). Les *entités* sont les modules fondamentaux de tous les modèles de simulation dans T-SIM. La communication entre les entités basées sur les événements est fournie par la classe `Event` de SSJ.

Il y a trois étapes dans la vie d'une entité : *la création, l'initialisation* et *la simulation*. L'objet représentant l'entité doit évidemment être créé dans le programme de simulation par un appel à l'instruction `new`. C'est la phase de la création.

Une fois celle-ci créée, l'entité doit être enregistrée dans la liste d'entités fournie par la méthode `TSim.getEntities`. Par la suite, le simulateur parcourt la liste d'entités enregistrées, pour exécuter leur méthode `initialise()`. C'est ici où une entité doit lire ses paramètres globaux et locaux, extraire l'information liée à la topolo-

gie du réseau et d'autres entités dans le système, ainsi qu'envoyer ses événements initiaux à la file d'attente d'événements.

Lorsque la phase d'initialisation est complétée, la méthode `TSim.start` est invoquée pour commencer la simulation. Dans cette phase, l'interaction entre les entités se réalise à travers des événements produits par la méthode `process(TSimEvent)`. Cette méthode indique toutes les propriétés concernant le comportement d'une nouvelle entité.

Pour récapituler, le constructeur doit fournir l'initialisation *minimale* de l'entité sans tenir compte des paramètres globaux ni locaux. Ensuite, la méthode `initialise()` est utilisée pour initialiser l'objet selon les paramètres globaux, locaux ou d'autres entités, dont la topologie du réseau. Finalement, la méthode `process(TSimEvent)` doit être remplacée pour donner lieu au codage décrivant le comportement de l'entité exigée par le modèle de simulation.

### A.2.2 Module OBS

La fonction principale du module OBS est de simuler le comportement d'un réseau OBS. Parmi les tâches à effectuer, nous y trouvons l'assemblage des rafales, la création de l'en-tête de contrôle, son acheminement à l'intérieur du réseau, la gestion de réservation de canaux à chaque noeud intermédiaire et finalement la mesure de la performance du réseau (voir figures A.6, A.7 et A.8).

Pour commencer à simuler un réseau nous devons d'abord générer des paquets. L'événement qui déclenche la production des paquets est généré par la classe `TsimEvent` en combinaison avec la classe `ObsPacketSource`. Les paquets produits sont ensuite envoyés au noeud attaché à la source

Lorsque l'en-tête de contrôle d'une rafale quelconque arrive à un noeud, le sys-

tème de réservation (défini par la classe abstraite `Pcu`) s'occupe de traiter la demande afin de trouver un canal au port de sortie du noeud. Si aucun canal n'est disponible, la rafale sera perdue. La première version de T-SIM implante les algorithmes Horizon, JET et S-JET dans les modules `PcuHorizon`, `PcuJet` et `PcuSjet`.

Nous allons utiliser le réseau OBS de la figure A.2, pour expliquer le fonctionnement de ce module. Nous pouvons identifier les composants principaux : les noeuds, les liens de fibre ainsi que les routes. Nous y trouvons également les sources ($\lambda$) produisant des paquets de données.

Pour effectuer une simulation, il faut créer d'abord un fichier décrivant le réseau. Ce fichier est mis comme argument de la ligne de commande au moment de lancer la simulation.

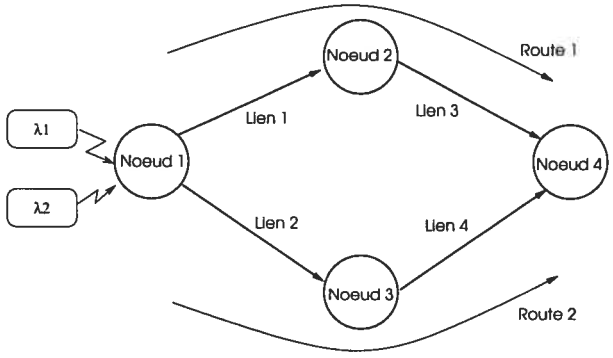



FIG. A.2 – Réseau exemple

À partir des informations contenues dans ce fichier, les entités faisant partie de la simulation (noeuds, liens, sources de paquets, etc.) sont créées et initialisées. Le fichier est sous-divisé en sections où chaque section comporte des paramètres locaux touchant le comportement de l'entité. Finalement, la dernière section s'occupe de la définition des paramètres globaux.

En utilisant comme modèle le réseau de la figure A.2, nous montrons le fichier contenant la description de ce réseau :

```
# définition de liens
# Type  No.  Id.  (ori, des) {options}
link 1 oc192 (1,2)
link 2 oc192 (1,3)
link 3 oc192 (2,4)
link 4 oc192 (3,4){wavelengths = 8}

# définition des noeuds
# type No. Id    {options}
node 1 obs_node
node 2 obs_node
node 3 obs_node
node 4 obs_node

# définition des sources de trafic
# type No. Id       {options}
entity 5 obs_source { target = 1, arrival rate = 40000.0}
entity 6 obs_source { target = 1, arrival rate = 50000.0}

# définition des routes
route_set unique =
{   1 : (1,2)
    2 : (3,4)
}

# Parameters globaux
parameters =
{  number of class = 1,
   pcu = "sjet",
   route set = "unique",
   wavelengths = 4,
   processing delay = 1.0E-6,
   burst duration = 5.0E-6
}
end
```

### Définition des liens

Un lien de fibre relie deux noeuds. Chaque lien est à la fois subdivisé en plusieurs sous canaux (longueurs d'onde). Dans T-SIM, les liens sont unidirectionnels, c'est-à-dire, les données ne voyagent que dans un seul sens. Analysons la ligne suivante :

`link 4 oc192 (3,4){wavelengths = 8}`. Dans ce cas, 4 est le numéro identifiant le lien. C'est un entier unique. Ensuite, `oc192`, définit le type de lien. Dans la première version de T-SIM, cette information n'est pas utilisée à l'intérieur du simulateur. Pour sa part, `(3,4)`, représente le sens du lien, où (3) est le noeud de départ et (4) le noeud destination. Finalement, `{wavelengths = 8}`, est un paramètre local optionnel spécifiant le nombre de longueurs d'onde pour ce lien. Si ce paramètre n'est pas inclu, c'est la valeur de `wavelengths` dans la section des paramètres globaux qui est utilisée.

### Définition des noeuds

Dans T-SIM, un commutateur est représenté par un noeud. La fonction de base de chaque noeud est de traiter les demandes de réservation des en-têtes de contrôle (`ControlPacket`). Le simulateur définit trois types de noeuds : le noeud de base (`ObsNodeEntity`), le noeud pour l'assemblage de rafales (`ObsEdgeRouter`) ainsi que le noeud TCP. Dans l'exemple ci haut, nous définissons quatre noeuds étiquetés de 1 à 4. Le type de noeud associé à l'identificateur `obs_node` se trouve dans le fichier en charge de lancer la simulation (`ObsMain.java`). Le simulateur permet aussi la définition de cinq paramètres optionnels : le temps de traitement, la longueur (en unité de temps) de la rafale, un temporisateur, l'algorithme de réservation à utiliser par le noeud et finalement la valeur (en unités du temps) du décalage.

### Définition des sources

Les sources sont responsables de fournir des paquets aux noeuds reliés à celles-ci. T-SIM définit quatre types de sources : la source `ObsPacketSource`, produisant des paquets pour être assemblés en rafales dans un noeud de type `ObsEdgeRouter`. La source `ObsBurstSource` qui génère directement des rafales sans avoir besoin

de les assembler. La source `ObsTraceSource` qui génère des paquets en prenant les informations sur le délai et le nombre d'octets à partir d'un fichier d'entrée. Finalement, la source `ObsSource`, une classe abstraite dont héritent les trois sources précédentes. Pour la première version de T-SIM, le processus d'arrivée des rafales par défaut est un processus de Poisson. Pour les versions ultérieures, nous offrirons d'autres types de processus d'arrivée.

Nous pouvons également ajouter à la définition de la source des paramètres optionnels pour mieux simuler son comportement. La première version de T-SIM définit cinq paramètres : le numéro de noeud auquel est attaché la source (`target`), la classe des paquets produits par la source (`class type`), la route à emprunter pour les paquets (`route`), le taux d'arrivée des paquets (`arrival rate`), et la durée moyenne de paquets pour cette source (`average packet duration`).

Dans la ligne : `entity 5 obs_source { target = 1, arrival rate = 40000.0)`, l'identificateur `obs_source` sera associé à la source définie dans le fichier `ObsMain.java`. Les paquets produits sont envoyés au noeud 1 à un taux d'arrivée de 40,000 paquets par seconde.

### Définition des routes

Dans T-SIM, une route est une séquence de liens formant le chemin à emprunter par la rafale. En utilisant comme exemple le réseau de la figure A.2, nous observons la présence de deux routes : la route 1 comprenant les liens 1 et 3 ainsi que la route 2 comprenant les liens 2 et 4. Le fichier de définition du réseau décrit cette information de la façon suivante :

```
route_set unique =
{   1 : (1,2)
    2 : (3,4)
}
```

**Paramètres Globaux**

- `arrival rate` : le nombre de paquets par seconde (taux d'arrivée).
- `average packet duration` : la durée moyenne de paquets.
- `burst duration` : la durée d'une rafale selon la classe.
- `pcu` : le type d'algorithme utilisé pour l'allocation de canaux.
- `offset` : la valeur à ajouter au décalage par défaut.
- `processing delay` : le temps que prend un noeud pour allouer un canal.
- `time out` : le temps maximum que prend l'assemblage d'une rafale (par défaut, cette option est désactivée).
- `wavelengths` : le nombre de longueurs d'onde par lien.

### A.2.3 Exécution d'une simulation

À partir de la ligne de commandes, il suffit de taper :

`java ObsMain.java <fichier_entrée> <durée_simulation> > <fichier_sortie>`

où *fichier_entrée* correspond au fichier qui contient la description du réseau à simuler et *fichier_sortie* au fichier pour enregistrer les résultats de la simulation. La durée de la simlation est donnée en secondes.

Le fichier ObsMain.java efectue les tâches suivantes : il commence par définir les types de sources, de noeuds ainsi que l'algorithme de réservation à utiliser par le réseau :

```
ObsEdgeRouter n = new ObsEdgeRouter (-1, "obs_node");
ObsBurstSource s = new ObsBurstSource (-1, "obs_source");
PcuHorizon h = new PcuHorizon (4);
```

Ensuite, il procéde à l'enregistrement des entités ainsi qu'à la route à emprunter par les rafales dans le simulateur. Dans cette étape, le simulateur parcourt la liste

des entités pour exécuter leur méthode `initialise`. Finalement, la simulation est lancée avec la commande `TSim.start()`. Le contenu du fichier `ObsMain.java` est illustré ici :

```
import umontreal.iro.lecuyer.simevents.*;
import umontreal.iro.lecuyer.rng.RandMrg;

public class ObsMain
{
  public static void main (String[] args)
  {
  if (args.length > 0)
    { EntityFactory factory = TSim.getEntityFactory ();

ObsEdgeRouter n = new ObsEdgeRouter (-1, "obs_node");
ObsBurstSource s = new ObsBurstSource (-1, "obs_source");
PcuHorizon h = new PcuHorizon (4);

factory.add (n, n.getType ());
factory.add (s, s.getType ());
factory.add (h, "horizon");

Entity re = new RoutingEntityStatic (Integer.MAX_VALUE,
            "routing_entity_static");
TSim.getEntities ().add (re);

TSim.createGraph (args[0]);

TSim.init ();

new EndOfSim().schedule(simulationTime);

TSim.start ();
    }
  }
}
```

À la fin de la simulation le fichier de sortie contient une description des rafales créés par noeud ainsi que le nombre de rafales allouées et perdues par noeud, par

lien et par route. La présentation de ces résultats peut être adaptée aux besoins des utilisateurs en modifiant le fichier `ObsStats.java`.

```
Node Id  Burst      Burst     Drop /
         Allocated  Drop      Allocated
1        4439       56        1,262E-2
2        2221       0         0,000E0
3        2216       0         0,000E0

Link Id  Burst      Burst     Drop /       Link Load
         Allocated  Drop      Allocated
1        2222       27        1,215E-2     2,222E-2
2        2217       29        1,308E-2     2,217E-2
3        2221       0         0,000E0      2,221E-2
4        2216       0         0,000E0      2,216E-2

Route Id  Burst      Burst     Drop /       Route Load
          Created    Drop      Created
1         2249       27        1,201E-2     2,249E-2
2         2246       29        1,291E-2     2,246E-2


Packet of type 0  created  89900
Total packet created  89900
Packet of type 0  rejected  1120  rejected/created = 1,246E-2
Total packet rejected  1120  rejected/created = 1,246E-2

Burst of type 0 created   4495
Total Burst created  4495
Burst of type 0  allocated  8876
Total Burst allocated  8876
Burst Class 0 blocked  56 BP = 1,246E-2
Total Burst rejected  56  rejected/created = 1,246E-2


REPORT on Tally stat. collector ==> Burst rejected
         min        max      average    standard dev.  num. obs
       3.000      8.000      5.600        1.506           10
  95.0% confidence interval for mean: (     4.523,     6.677 )

Simulation execution time = 0.222
```

## A.3 Conclusion

T-SIM est un simulateur conçu pour les réseaux optiques. Il utilise les outils de l'environnement de simulation stochastique SSJ pour la gestion des événements et la génération des nombres aléatoires. La première version de T-SIM est formée de deux modules : le noyau du simulateur ainsi que le module OBS. L'idée derrière sa conception est de fournir un outil de simulation pratique qui peut être adapté selon les besoins des utilisateurs.

Bien que cette première version ne considère que les réseaux OBS, l'intégration de modules additionnels permettra d'augmenter la flexibilité de T-SIM pour simuler d'autres types de réseaux ainsi que des protocoles de communication et de routage.

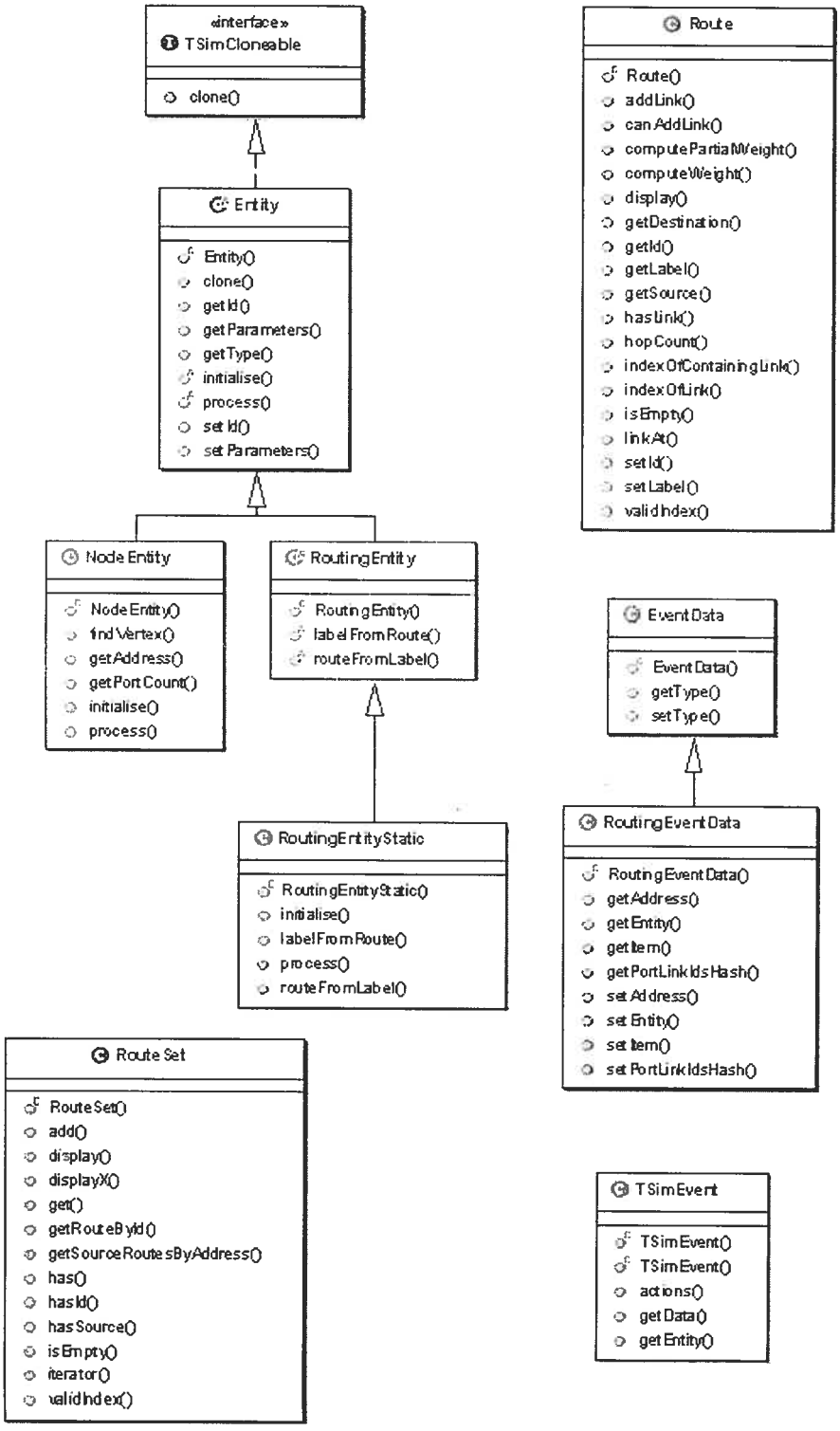



FIG. A.3 – Classes T-SIM (1)

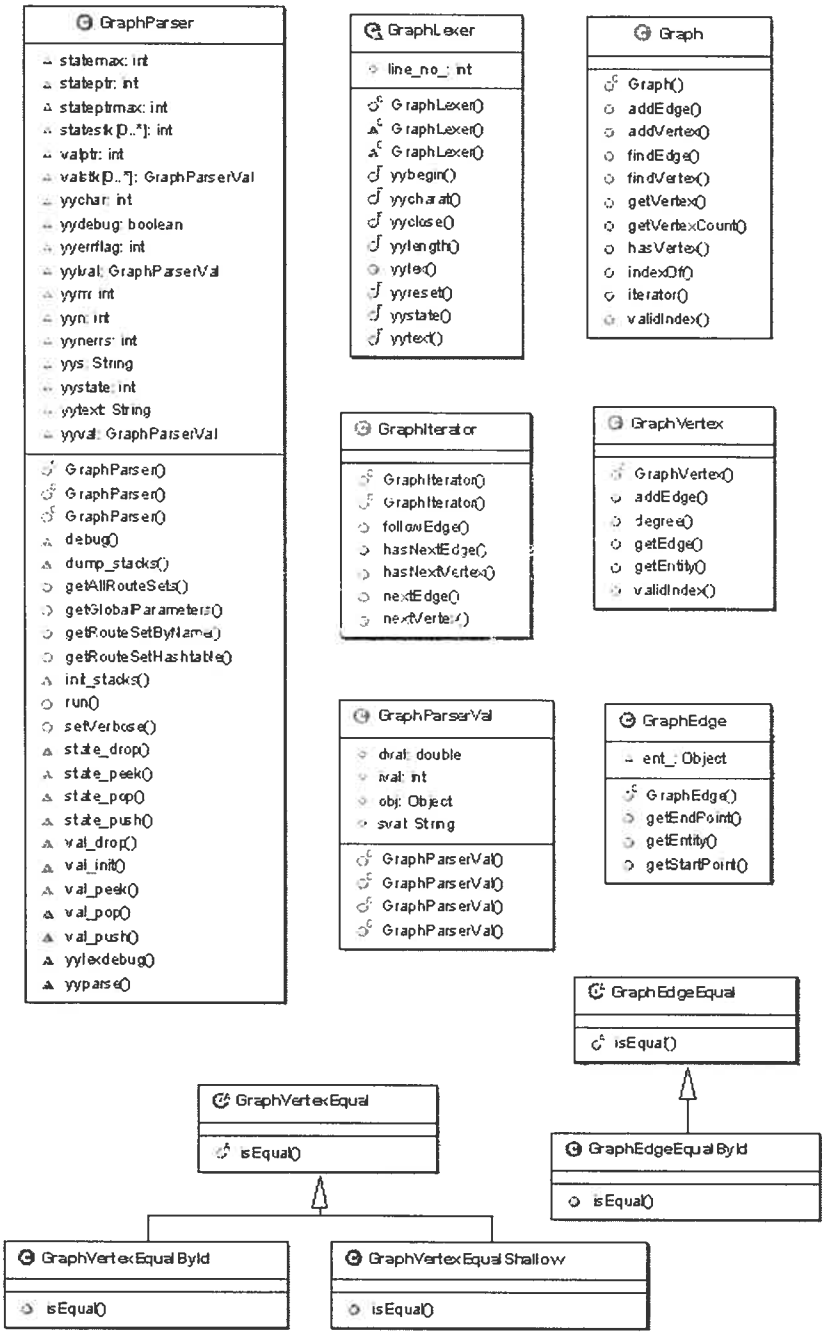



FIG. A.4 – Classes T-SIM (2)

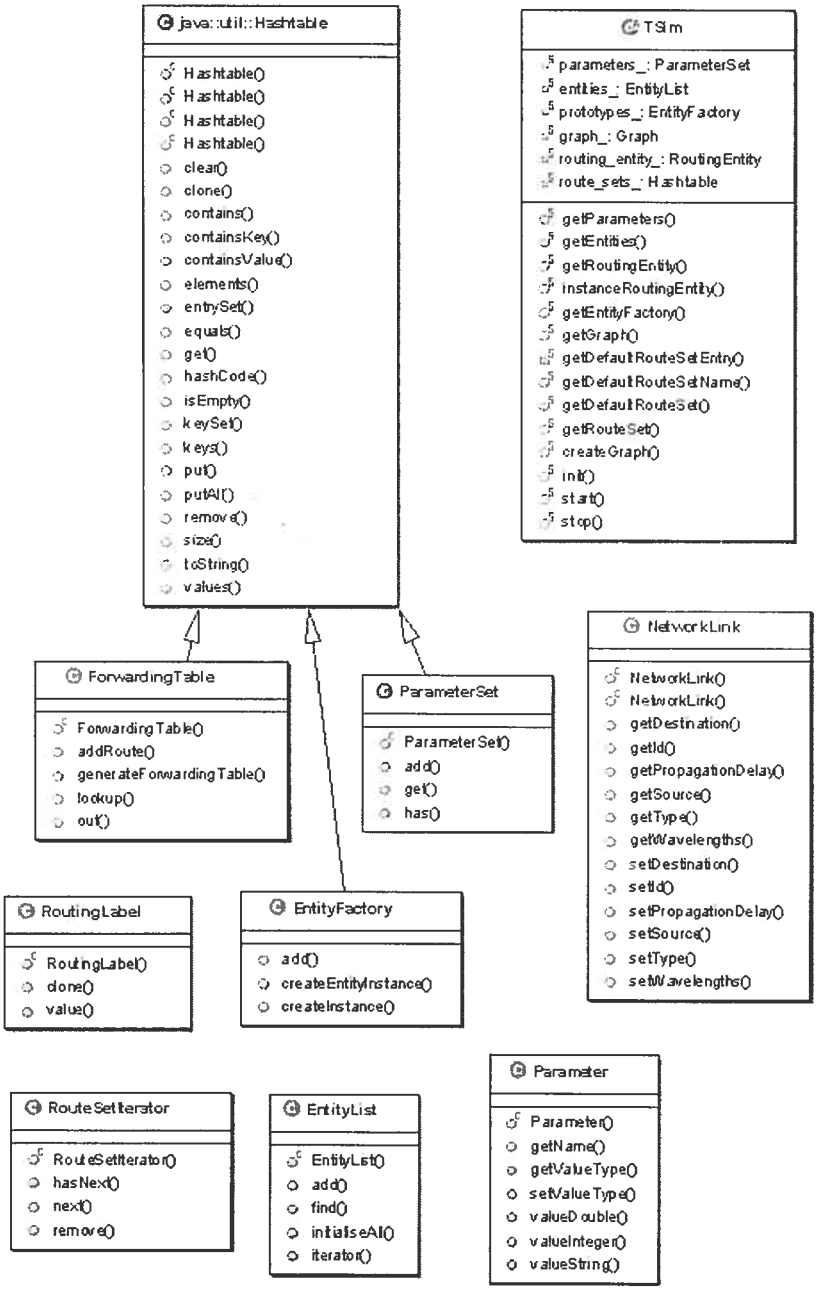



FIG. A.5 – Classes T-SIM (3)

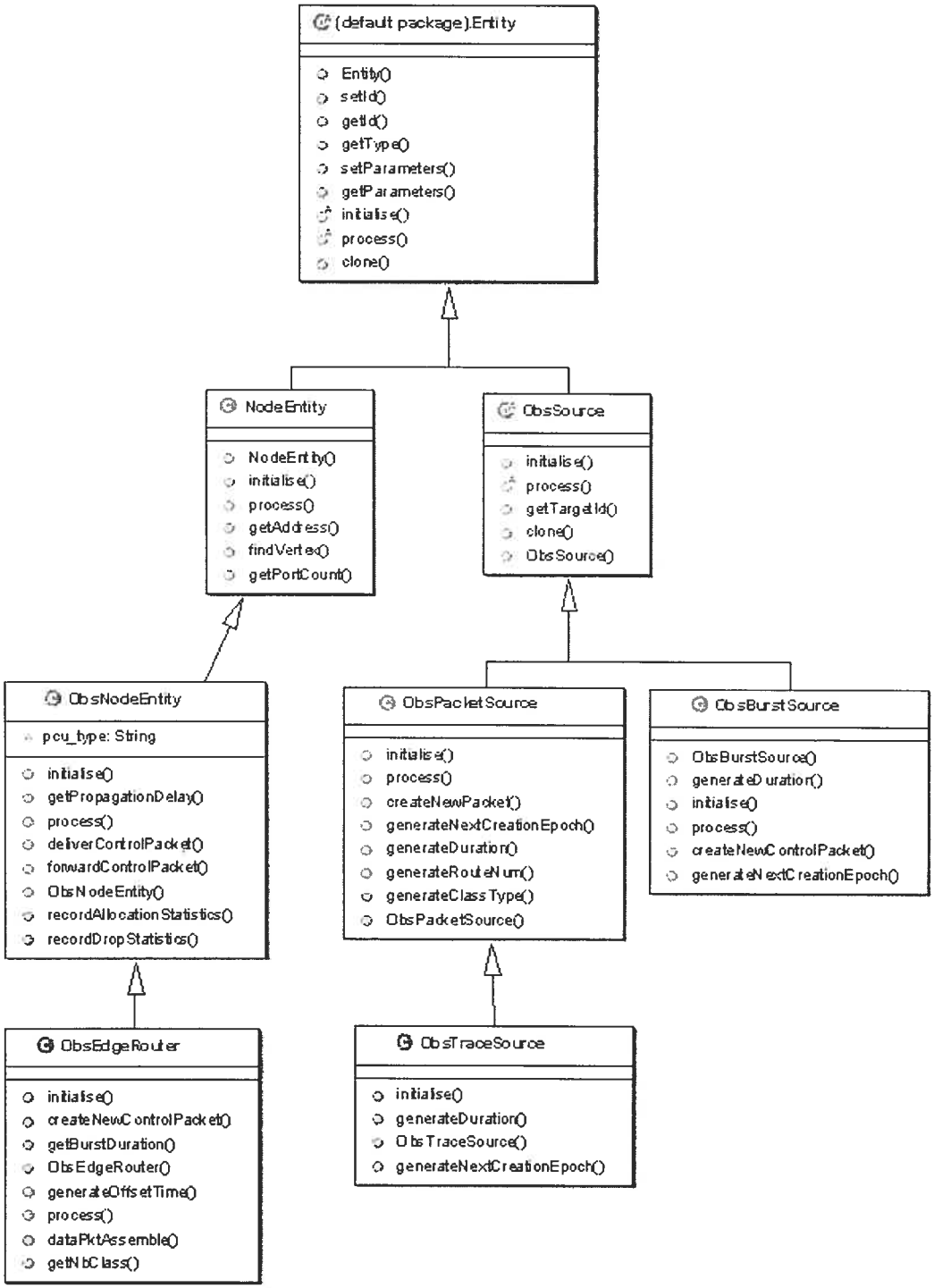



FIG. A.6 – Classes OBS (1)

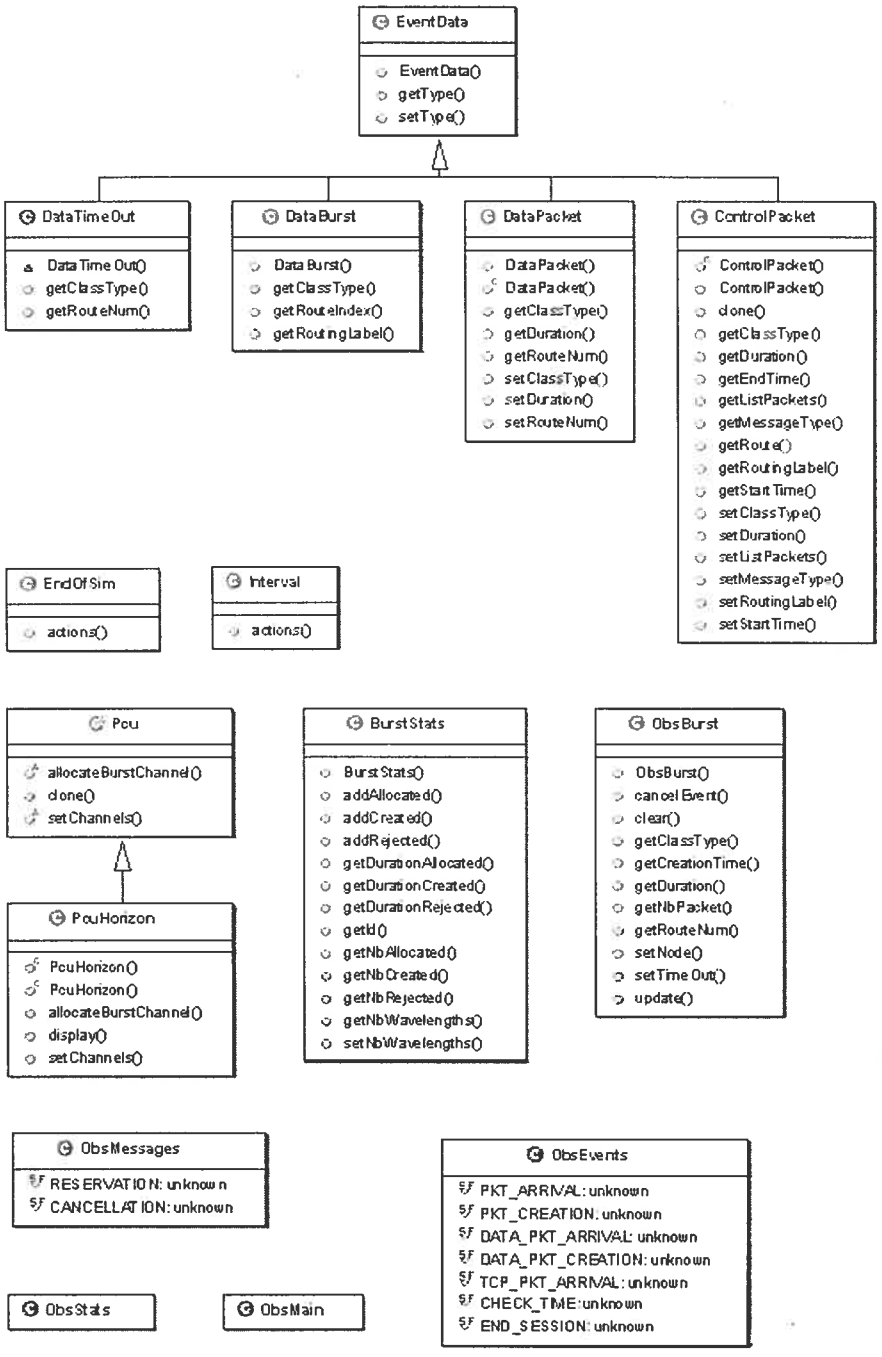



FIG. A.7 – Classes OBS (2)

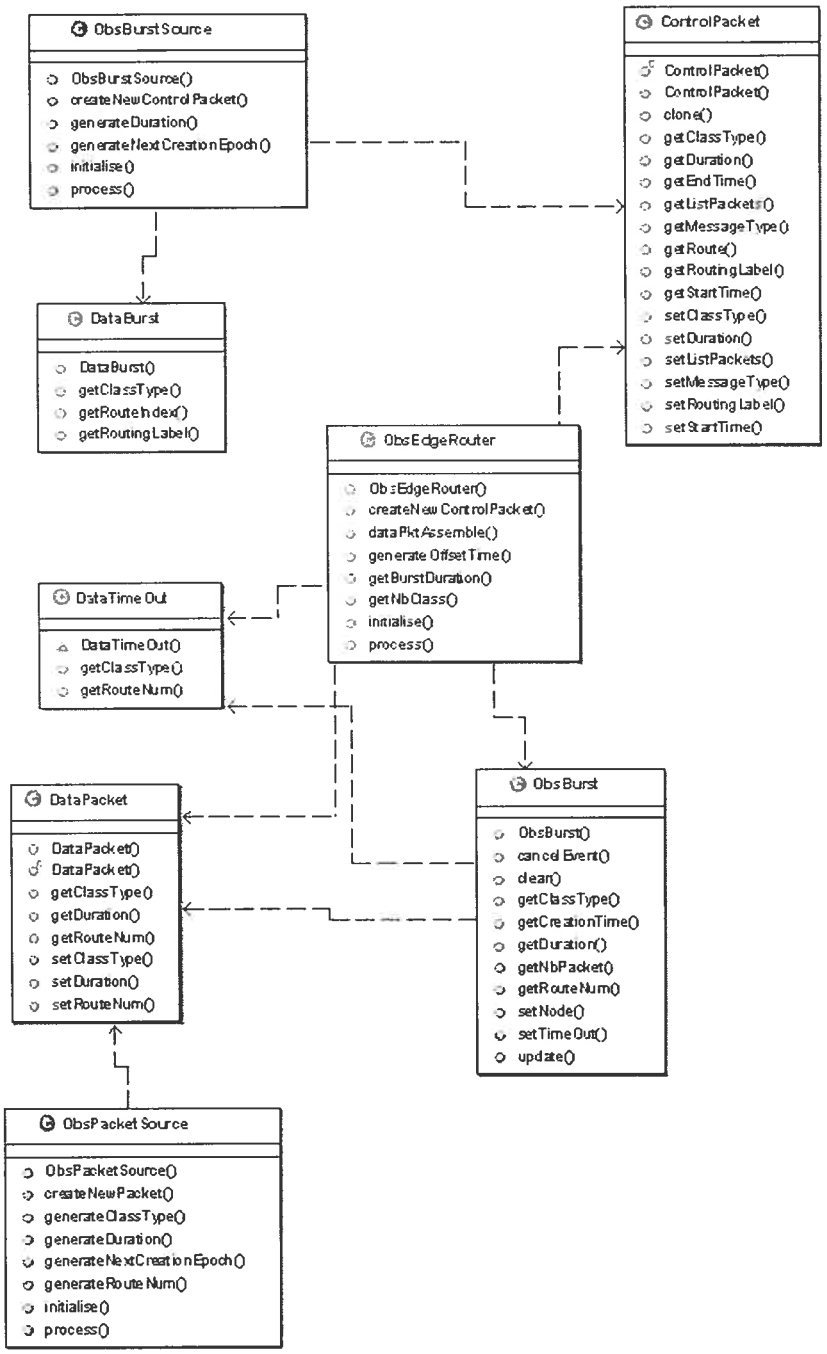



FIG. A.8 – Classes OBS (3)